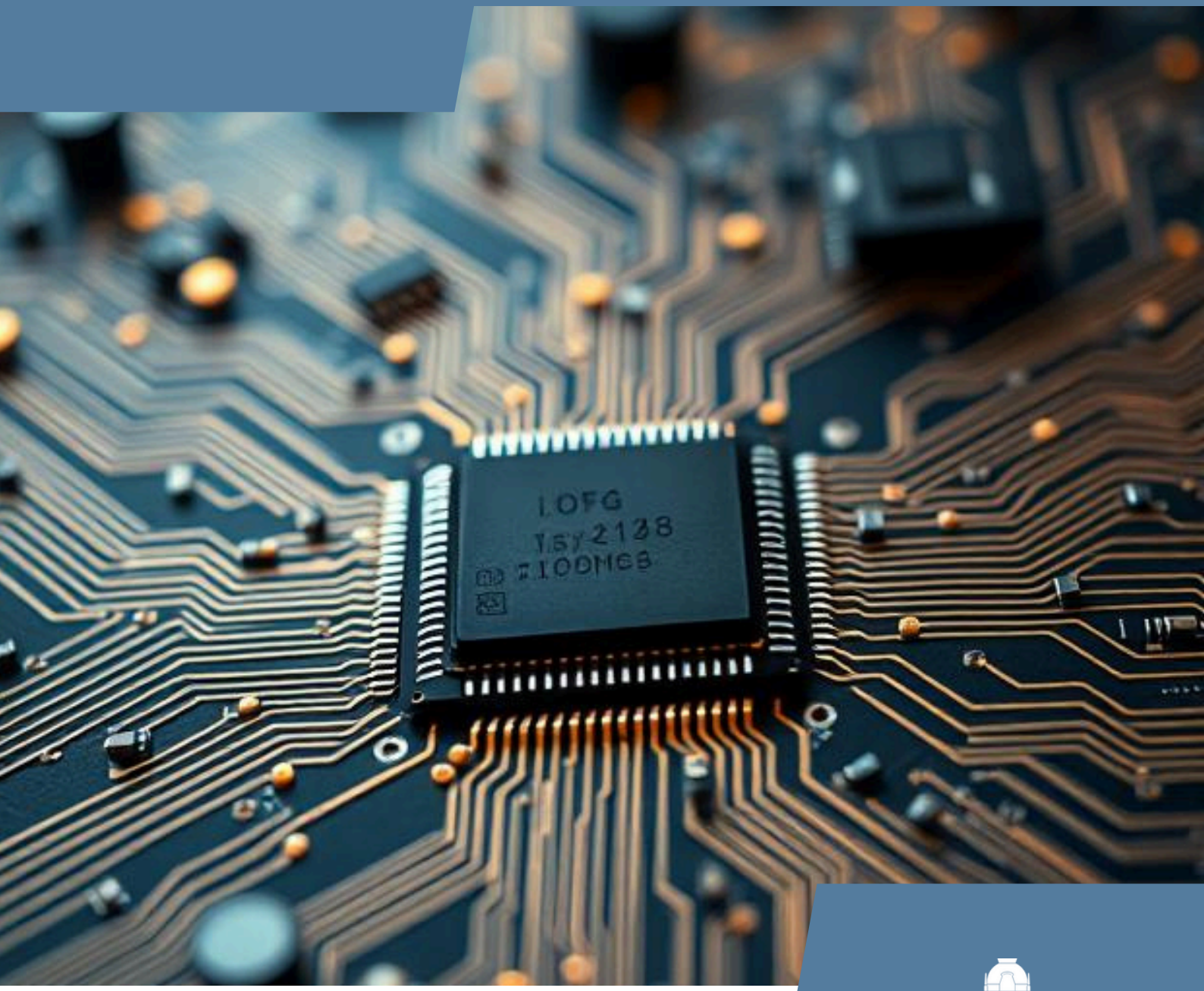
MIKRORAČUNALNIŠKE ARHITEKTURE

Zbirka rešenih nalog



Zmago Brezočnik

Univerzitetna založba
Univerze v Mariboru

University of Maribor

Faculty of Electrical Engineering
and Computer Science

Mikroračunalniške arhitekture

Zbirka rešenih nalog

Avtor
Zmago Brezočnik

September 2025

Naslov **Mikroračunalniške arhitekture**
Title *Microcomputer Architectures*

Podnaslo **Zbirka rešenih nalog**
Subtitle *Collection of Solved Problems*

Avtor Zmago Brezočnik
Author (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)

Recenzija Tatjana Kapus
Review (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)

Lektoriranje AMIDAS d.o.o.
Language editing

Tehnična urednika Zmago Brezočnik
Technical editors (Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko)

Jan Perša
(Univerza v Mariboru, Univerzitetna založba)

Oblikovalec ovitka Špela Brezočnik
Cover designer

Grafika na ovitku DeepAI, foto: Deep AI, Inc., 2025
Cover graphics

Založnik **Univerza v Mariboru**
Published by **Univerzitetna založba**
Slomškov trg 15, 2000 Maribor, Slovenija
https://press.um.si, zalozba@um.si

Izdajatelj **Univerza v Mariboru**
Issued by **Fakulteta za elektrotehniko, računalništvo in informatiko**
Koroška cesta 46, 2000 Maribor, Slovenija
https://www.feri.um.si, feri@um.si

Izdaja 1. izdaja
Edition

Izdano Maribor, Slovenija, September, 2025
Published at

Vrsta publikacije E-knjiga
Publication type

Dostopno na https://press.um.si/index.php/ump/catalog/book/1055
Available at

CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004.2(076.2)(0.034.2)

BREZOČNIK, Zmago
Mikroračunalniške arhitekture : zbirka rešenih nalog / avtor Zmago Brezočnik. - 1st ed. - E-knjiga. - Maribor : Univerza v Mariboru, Univerzitetna založba, 2025
Način dostopa (URL):

https://press.um.si/index.php/ump/catalog/book/1055

ISBN 978-961-299-054-1 (PDF)
doi: 10.18690/um.feri.10.2025
COBISS.SI-ID 249274627

ISBN 978-961-299-054-1 (pdf)

DOI https://doi.org/10.18690/um.feri.10.2025

Cena / *Price* Brezplačni izvod

Odgovorna oseba založnika / *For publisher* Prof. dr. Zdravko Kačič, rektor Unvierze v Mariboru

Attribution Brezočnik, Z. (2025). *Mikroračunalniške arhitekture: Zbirka rešenih nalog.* Univerza v Mariboru, Univerzitetna založba. doi: 10.18690/um.feri.10.2025

Vsebina

Seznam uporabljenih kratic

Kratica	Angleški pomen	Slovenski pomen
AD	Analog-to-Digital	analogno-digitalni(-a)
ALE	Arithmetic and Logic Unit	aritmetična in logična enota
B	byte	bajt (2^3 bitov)
BCD	Binary Coded Decimal	dvojiško kodirano desetiško število
BE	Big Endian	veliki endian
bps	bits per second	biti na sekundo (b/s)
CISC	Complex Instruction Set Computer	računalnik s širokim naborom ukazov
CPE	Central Processing Unit	centralna procesna enota
CPI	Clocks (Cycles) per Instruction	število urinih ciklov na ukaz
cps	characters per second	število znakov na sekundo
DMA	Direct Memory Access	neposredni dostop do pomnilnika
DMAC	DMA Controller	krmilnik DMA
DRAM	Dynamic Random Access Memory	dinamični RAM
EABI	Embedded Appl. Binary Interface	binarni vmesnik za vgrajene aplikacije
EDAC	Error Detection and Correction	odkrivanje in popravljanje napak
EIA	Electronic Industries Association	Združenje elektronske industrije
ER	Effective data Rate	dejanska hitrost prenosa podatkov
FIT	Failures in Time	število odpovedi v času 10^9 ur
GB	gigabyte	gigabajt (2^{30} bajtov)
IRA	International Reference Alphabet	mednarodna referenčna abeceda
kB/s	kilobyte per second	kilobajt na sekundo (10^3 B/s)
kb/s	kilobit per second	kilobit na sekundo (10^3 b/s)
KB	kilobyte	kilobajt (2^{10} bajtov)
LE	Little Endian	mali endian
LRU	Least Recently Used	že najdlje časa neuporabljen
MB	megabyte	megabajt (2^{20} bajtov)
MIPS	Million Instructions Per Second	milijonov ukazov na sekundo
MMX	Multimedia Extension	razširitev za multimedijo
MSI	Mean Scale Integration	srednja stopnja integracije
MTBF	Mean Time Between Failures	povprečni čas med odpovedima
NASM	Netwide Assembler for the x86	zbirnik za arhitekturo Intel x86
PIT	Programmable Interval Timer	programirljivi intervalni časovnik
RAID	Redundant Array of Independed Disks	redundantno polje neodvisnih diskov
RAM	Random Access Memory	pomnilnik z naključnim dostopom
RISC	Reduced Instruction Set Computer	računalnik z zoženim naborom ukazov
rpm	revolutions per minute	obrati na minuto
SCC	Serial Communication Controller	serijski komunikacijski krmilnik
SSI	Small Scale Integration	nizka stopnja integracije
TB	terabyte	terabajt (2^{40} bajtov)
V/I	Input/Output	vhodno-izhodni(-a)
x86	Intel architecture based on 8086	Intelova arhitektura na osnovi 8086

Predgovor

Zbirka je nastala na podlagi dolgoletnega pedagoškega dela pri več predmetih s področja mikroračunalniških arhitektur, mikroprocesorjev in mikrokrmilnikov na Fakulteti za elektrotehniko, računalništvo in informatiko Univerze v Mariboru. Namenjena je študentom kot pripomoček pri pripravi na kolokvije in pisne izpite z računskimi nalogami. Vsebuje 170 nalog z rešitvami in s celotnim potekom reševanja. Bralec naj poskuša reševati naloge čim bolj samostojno, rešitve pa naj uporabi zgolj za preveritev lastnega znanja. Vsebina večine nalog izhaja iz temeljnih učbenikov [1, 2, 3], iz referenčnega gradiva o mikroprocesorjih z arhitekturo Intel x86 [4] ter iz vira [5] o programiranju procesorjev družine ARM Cortex-M4. Pri vsaki izmed teh nalog je sklic na vir. Nekatere zahtevnejše naloge so označene z zvezdico (*). Njihova težavnost praviloma presega raven nalog s pisnega dela izpita in so namenjene študentom, ki želijo svoje znanje še dodatno poglobiti.

Naloge so razporejene v deset poglavij. Po kratkem uvodu sledi drugo poglavje, namenjeno ocenjevanju zmogljivosti računalnikov. V tretjem poglavju so naloge, ki obravnavajo komunikacijo med osnovnimi komponentami računalnika. Četrto poglavje vsebuje naloge, namenjene snovanju in izračunu parametrov predpomnilnika. Notranji pomnilnik, zlasti odkrivanje in popravljanje napak s Hammingovimi kodi, obravnavajo naloge petega poglavja, zunanji pomnilnik pa naloge šestega. Sedmo poglavje je posvečeno vhodno-izhodni organizaciji mikroprocesorjev. Osmo poglavje prinaša naloge za analizo učinkov izvajanja različnih strojnih ukazov mikroprocesorja. Načini naslavljanja in formati ukazov so obravnavani v nalogah devetega poglavja. Zadnje, deseto poglavje vključuje naloge za analizo kratkih odsekov kode in pisanje kratkih programov v zbirniku za mikroprocesorje arhitektur Intel x86 in ARM. Na koncu so za več besedilnih nalog priloženi programi v zbirniku oziroma v jeziku C in zbirniku za procesorje ARM Cortex-M4, razviti in preizkušeni v integriranem razvojnem okolju STM32CubeIDE. Zbirko zaključuje priloga A z dokazi štirih izrekov, uporabljenih v nalogah, in seznam virov.

Zbirka je namenjena predvsem študentom UN ŠP Elektrotehnika, smer Elektronika, in študentom UN ŠP Telekomunikacije pri predmetu *Mikroračunalniške arhitekture*. Namenjena je tudi študentom VS ŠP Elektrotehnika, smer Elektronika, pri predmetu *Mikroprocesorski sistemi I*. Za te skupine so posebej relevantna poglavja 1, 2, 3, 4, 8, 9 in 10. Študentom MAG ŠP Elektrotehnika, smer Elektronika, bosta pri predmetu *Mikroprocesorski sistemi* koristili predvsem poglavji 6 in 7, študentom VS ŠP Elektrotehnika, smer Elektronika, pri predmetu *Mikroprocesorski sistemi II* pa poglavji 5 in 7.

1 Uvod

1.1 Razlika med organizacijo in arhitekturo računalnika

Kakšna je v splošnem razlika med organizacijo in arhitekturo računalnika [1]?

Rešitev:
Arhitektura računalnika se nanaša na tiste atribute sistema, ki so vidni programerju, ali povedano z drugimi besedami, na tiste atribute, ki imajo neposredni vpliv na logično izvajanje programa. *Organizacija računalnika* se nanaša na operativne enote in njihove medsebojne povezave, ki izvršujejo arhitekturne specifikacije. Primeri arhitekturnih atributov vključujejo nabor ukazov, število bitov, uporabljenih za predstavitev različnih podatkovnih tipov (npr. števil, znakov), vhodne/izhodne (V/I) mehanizme in tehnike za naslavljanje pomnilnika. Organizacijski atributi vključujejo tiste podrobnosti strojne opreme, ki so programerju nevidne, kot so na primer krmilni signali, vmesniki med računalnikom in perifernimi napravami ter uporabljena tehnologija pomnilnika.

1.2 Razlika med strukturo in funkcijo računalnika

Kakšna je v splošnem razlika med strukturo in funkcijo računalnika [1]?

Rešitev:
Struktura računalnika se nanaša na način, kako so komponente računalnika med seboj povezane. *Funkcija računalnika* se nanaša na delovanje vsake posamezne komponente kot dela strukture.

1.3 Glavne funkcije računalnika

Katere so štiri glavne funkcije računalnika [1]?

Rešitev:
Štiri glavne funkcije računalnika so procesiranje podatkov, hramba podatkov, premikanje podatkov in nadzor.

1.4 Glavne strukturne komponente računalnika

Naštejte in na kratko definirajte glavne strukturne komponente računalnika [1].

Rešitev:
Glavne strukturne komponente računalnika so:

- *Centralna procesna enota (CPE).* Nadzira delovanje računalnika in izvaja njegove funkcije procesiranja podatkov. Pogosto ji rečemo kar procesor.
- *Glavni pomnilnik.* Hrani podatke in programe.
- *V/I.* Premika podatke med računalnikom in zunanjim okoljem.

- *Medsebojne povezave.* To je nek mehanizem, ki preskrbi komunikacijo med CPE, glavnim pomnilnikom in V/I. Običajen primer medsebojne povezave komponent sistema je sistemsko vodilo, ki je sestavljeno iz množice prevodnih vezic, na katere so povezane vse druge komponente.

1.5 Glavne strukturne komponente procesorja

Naštejte in na kratko definirajte glavne strukturne komponente procesorja [1].

Rešitev:

- *Krmilna enota.* Krmili operacije CPE in s tem računalnika.
- *Aritmetična in logična enota (ALE).* Izvaja računalnikove funkcije procesiranja podatkov.
- *Registri.* Poskrbijo za hrambo informacije znotraj CPE.
- *Medsebojne povezave v CPE.* To je nek mehanizem, ki preskrbi komunikacijo med krmilno enoto, ALE in registri.

2 Zmogljivost računalnikov

2.1 Ocenjevanje zmogljivosti računalnikov (1/7)

Primerjalni program izvajamo na procesorju s frekvenco 40 MHz. Izvedeni program sestavlja 100.000 ukazov z naslednjo ukazno mešanico in številom urinih ciklov:

Tip ukaza	**Število ukazov**	**Število urinih ciklov na ukaz**
Celoštevilčni aritmetični	45.000	1
Prenos podatkov	32.000	2
Plavajoča vejica	15.000	2
Prenos nadzora	8.000	2

Določite število urinih ciklov na ukaz (CPI), zmogljivost v MIPS (število izvedenih ukazov v milijonih na sekundo) in čas izvajanja za ta program [1].

Rešitev:

$$CPI = \frac{\sum_i CPI_i \cdot I_i}{I_C} = \frac{(1 \cdot 45 + 2 \cdot 32 + 2 \cdot 15 + 2 \cdot 8) \cdot 10^3}{(45 + 32 + 15 + 8) \cdot 10^3} = 1{,}55$$

$$\text{zmogljivost v MIPS} = \frac{f}{CPI \cdot 10^6} = \frac{40 \cdot 10^6}{1{,}55 \cdot 10^6} = 25{,}8$$

$$T = \frac{I_C \cdot CPI}{f} = \frac{100 \cdot 10^3 \cdot 1{,}55}{40 \cdot 10^6} = 3{,}87\,\text{ms}$$

2.2 Ocenjevanje zmogljivosti računalnikov (2/7)

Imamo dva različna računalnika z dvema različnima ukaznima množicama. Oba računalnika imata uro frekvence 200 MHz. V spodnji tabeli so zbrane meritve z obeh računalnikov ob izvajanju dane množice primerjalnih programov.

Tip ukaza	**Število ukazov (v milijonih)**	**Število ciklov na ukaz**
Računalnik A		
Aritmetični in logični	8	1
Nalaganje in shranjevanje	4	3
Vejanje	2	4
Drugi	4	3
Računalnik B		
Aritmetični in logični	10	1
Nalaganje in shranjevanje	8	2
Vejanje	2	4
Drugi	4	3

Določite CPI, zmogljivost v MIPS in čas izvajanja za vsak računalnik. Rezultate komentirajte [1].

Rešitev:

$$CPI_A = \frac{\sum_i CPI_i \cdot I_i}{I_C} = \frac{(8 \cdot 1 + 4 \cdot 3 + 2 \cdot 4 + 4 \cdot 3) \cdot 10^6}{(8 + 4 + 2 + 4) \cdot 10^6} \approx 2{,}22$$

$$\text{zmogljivost v MIPS}_\text{A} = \frac{f}{CPI_A \cdot 10^6} = \frac{200 \cdot 10^6}{2{,}22 \cdot 10^6} = 90$$

$$T_A = \frac{I_C \cdot CPI_A}{f} = \frac{18 \cdot 10^6 \cdot 2{,}22}{200 \cdot 10^6} = 0{,}2\,\text{s}$$

$$CPI_B = \frac{\sum_i CPI_i \cdot I_i}{I_C} = \frac{(10 \cdot 1 + 8 \cdot 2 + 2 \cdot 4 + 4 \cdot 3) \cdot 10^6}{(10 + 8 + 2 + 4) \cdot 10^6} \approx 1{,}92$$

$$\text{zmogljivost v MIPS}_\text{B} = \frac{f}{CPI_B \cdot 10^6} = \frac{200 \cdot 10^6}{1{,}92 \cdot 10^6} = 104$$

$$T_B = \frac{I_C \cdot CPI_B}{f} = \frac{24 \cdot 10^6 \cdot 1{,}92}{200 \cdot 10^6} = 0{,}23\,\text{s}$$

Čeprav ima računalnik B večjo zmogljivost v MIPS kot računalnik A, je čas izvajanja iste množice primerjalnih programov na njem daljši.

2.3 Ocenjevanje zmogljivosti računalnikov (3/7)

Primer zgodnjega računalnika tipa CISC (računalnik s širokim naborom ukazov) je VAX 11/780, primer zgodnjega tipa računalnika RISC (računalnik z zoženim naborom ukazov) pa IBM RS/6000. Uporaba tipičnega primerjalnega programa je dala naslednje rezultate:

Procesor	**Frekvenca ure**	**Zmogljivost**	**čas CPE**
VAX 11/780	5 MHz	1 MIPS	12x sekund
IBM RS/6000	25 MHz	18 MIPS	x sekund

Zadnji stolpec prikazuje, da je VAX zahteval za izvedbo programa 12-krat več časa CPE kot IBM [1].

a) Kolikšno je razmerje števila ukazov izvedljive strojne kode za ta primerjalni program med računalnikoma IBM RS/6000 in VAX 11/780?

b) Koliko znašata vrednosti CPI za oba računalnika?

Rešitev:

a) Zveza med številom ukazov programa, ki se izvaja na računalniku z zmogljivostjo v MIPS in za izvedbo potrebuje T (procesorskega) časa, je:

$$I_C = (\text{zmogljivost v MIPS}) \cdot 10^6 \cdot T$$

Razmerje števila ukazov med računalnikoma IBM RS/6000 in VAX 11/780 je:

$$\frac{I_{C\_IBM}}{I_{C\_VAX}} = \frac{18}{1} \cdot \frac{10^6}{10^6} \cdot \frac{x}{12x} = 1{,}5$$

b) Zveza med številom urinih ciklov na ukaz, frekvenco ure in mero MIPS je:

$$CPI = \frac{f}{(\text{zmogljivost v MIPS}) \cdot 10^6}$$

$$CPI_{VAX} = \frac{5\,\text{MHz}}{1\ \text{MIPS} \cdot 10^6} = \frac{5 \cdot 10^6}{1 \cdot 10^6} = 5$$

$$CPI_{IBM} = \frac{25\,\text{MHz}}{18\ \text{MIPS} \cdot 10^6} = \frac{25 \cdot 10^6}{18 \cdot 10^6} = 1{,}39$$

2.4 Ocenjevanje zmogljivosti računalnikov (4/7)

Če na nekem računalniku zaporedoma izvajamo n programov in njihove čase izvajanja označimo z x_1, x_2, …, x_n, za izračun srednje vrednosti časa izvajanja vseh n programov najpogosteje uporabimo aritmetično (R_A), harmonično (R_H) ali geometrijsko (R_G) srednjo vrednost. Izračunamo jih po naslednjih definicijah:

$$R_A := \frac{1}{n}\sum_{i=1}^{n} x_i, \quad R_G := \left(\prod_{i=1}^{n} x_i\right)^{1/n}, \quad R_H := \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Za poljuben n velja izrek $R_A \geq R_G \geq R_H$. Dokaz najdete v prilogi **A.1**. Samo za $n = 2$ pa velja izrek $R_A \cdot R_H = R_G^2$. Dokaz najdete v prilogi **A.2**.

Štiri primerjalne programe smo izvedli na treh različnih računalnikih in dobili naslednje rezultate:

	Računalnik A	**Računalnik B**	**Računalnik C**
Program 1	1	10	20
Program 2	1.000	100	20
Program 3	500	1.000	50
Program 4	100	800	100

Tabela prikazuje izvajalne čase v sekundah za 100.000.000 izvedenih ukazov v vsakem izmed štirih programov. Izračunajte zmogljivost v MIPS za vsak računalnik in vsak program. Nato izračunajte aritmetične in harmonične srednje vrednosti ter rangirajte računalnike na osnovi aritmetične srednje vrednosti in harmonične srednje vrednosti [1].

Rešitev:
Zmogljivost v MIPS izračunamo iz enačbe

$$\text{zmogljivost v MIPS} = \frac{I_C}{T \times 10^6} = \frac{100}{T}$$

	Računalnik A	**Računalnik B**	**Računalnik C**
Program 1	100	10	5
Program 2	0,1	1	5
Program 3	0,2	0,1	2
Program 4	1	0,125	1

	Aritmetična srednja vrednost	Rang	Harmonična srednja vrednost	Rang
Računalnik A	25,325	1	0,25	2
Računalnik B	2,8	3	0,21	3
Računalnik C	3,25	2	2,1	1

2.5 Ocenjevanje zmogljivosti računalnikov (5/7)

V spodnji tabeli so zbrani izvajalni časi v sekundah za pet različnih primerjalnih programov na treh računalnikih [1].

Primerjalni program	Računalnik		
	R	**M**	**Z**
Program 1	417	244	134
Program 2	83	70	70
Program 3	66	153	135
Program 4	39.449	35.527	66.000
Program 5	772	368	369

a) Izračunajte metriko hitrosti za vsak procesor za vsak primerjalni program, normalizirano na računalnik R. To pomeni, da so vse vrednosti razmerij za R enake 1,0. Druga razmerja r_i se izračunajo po enačbi

$$r_i = \frac{Tref_i}{Tsut_i}$$

kjer se za referenčni sistem vzame računalnik R. Nato izračunajte aritmetično srednjo vrednost za vsak sistem.

b) Ponovite točko **a)**, če za referenčni sistem izberete računalnik M.

c) Kateri računalnik je najpočasnejši pri obeh prejšnjih izračunih?

d) Ponovite izračune iz točk **a)** in **b)** z uporabo geometrične srednje vrednosti R_G, definirane z enačbo:

$$R_G := \left(\prod_{i=1}^{n} r_i \right)^{1/n}$$

Rešitev:

a) Normalizirano na računalnik R:

Primerjalni program	Računalnik		
	R	M	Z
Program 1	1,00	1,71	3,11
Program 2	1,00	1,19	1,19
Program 3	1,00	0,43	0,49
Program 4	1,00	1,11	0,60
Program 5	1,00	2,10	2,09
Aritmetična srednja vrednost	1,00	1,31	1,50

b) Normalizirano na računalnik M:

Primerjalni program	Računalnik		
	R	M	Z
Program 1	0,59	1,00	1,82
Program 2	0,84	1,00	1,00
Program 3	2,32	1,00	1,13
Program 4	0,90	1,00	0,54
Program 5	0,48	1,00	1,00
Aritmetična srednja vrednost	1,01	1,00	1,10

c) Spomnimo, da večji kot je r_i, večja je hitrost testiranega sistema. Na osnovi rezultatov v **a)** je R znatno najpočasnejši računalnik. Na osnovi rezultatov v **b)** je M za malenkost najpočasnejši računalnik.

d) Normalizirano na računalnik R in nato na računalnik M:

Primerjalni program	Računalnik		
	R	M	Z
Program 1	1,00	1,71	3,11
Program 2	1,00	1,19	1,19
Program 3	1,00	0,43	0,49
Program 4	1,00	1,11	0,60
Program 5	1,00	2,10	2,09
Geometrijska srednja vrednost	1,00	1,15	1,18

Primerjalni program	Računalnik		
	R	M	Z
Program 1	0,59	1,00	1,82
Program 2	0,84	1,00	1,00
Program 3	2,32	1,00	1,13
Program 4	0,90	1,00	0,54
Program 5	0,48	1,00	1,00
Geometrijska srednja vrednost	0,87	1,00	1,02

Z uporabo geometrične srednje vrednosti je najpočasnejši računalnik R, ne glede na to, kateri računalnik se uporablja za normalizacijo.

2.6 Ocenjevanje zmogljivosti računalnikov (6/7)

Da bi si razjasnili rezultate prejšnje naloge, si poglejmo preprostejši primer [1].

Primerjalni program	**Računalnik**		
	X	**Y**	**Z**
1	20	10	40
2	40	80	20

a) Izračunajte aritmetično srednjo vrednost za vsak sistem ob uporabi računalnika X za referenčni sistem in nato ob uporabi računalnika Y za referenčni sistem. Argumentirajte, da imajo vsi trije računalniki intuitivno približno ekvivalentne zmogljivosti in da aritmetična srednja vrednost daje zavajajoče rezultate.

b) Izračunajte geometrično srednjo vrednost za vsak sistem ob uporabi računalnika X za referenčni sistem in nato ob uporabi računalnika Y za referenčni sistem. Argumentirajte, da so rezultati bolj realistični kot z uporabo aritmetične srednje vrednosti.

Rešitev:

a) Normalizirano na računalnik X:

Primerjalni program	**Računalnik**		
	X	**Y**	**Z**
1	1	2,0	0,5
2	1	0,5	2,0
Aritmetična srednja vrednost	1	1,25	1,25
Geometrijska srednja vrednost	1	1	1

Normalizirano na računalnik Y:

Primerjalni program	**Računalnik**		
	X	**Y**	**Z**
1	0,5	1	0,25
2	2,0	1	4,0
Aritmetična srednja vrednost	1,25	1	2,125
Geometrijska srednja vrednost	1	1	1

Računalnik Y je dvakrat hitrejši od računalnika X za primerjalni program 1, toda samo pol toliko hiter za primerjalni program 2. Podobno je računalnik Z pol toliko močan kot X za primerjalni test 1, vendar dvakrat hitrejši za primerjalni test 2. Intuitivno imajo ti trije računalniki ekvivalentne zmogljivosti.

Vendar pa, če normaliziramo na X in računamo aritmetično srednjo vrednost hitrostne metrike, ugotovimo, da sta Y in Z 25 % hitrejša kot X. Če zdaj normaliziramo na Y in izračunamo aritmetično srednjo vrednost metrike hitrosti, ugotovimo, da je X 25 % hitrejši od Y in Z je več kot dvakrat hitrejši od Y. Očitno je, da je aritmetična srednja vrednost v tem kontekstu brez vrednosti.

b) Če uporabimo geometrijsko srednjo vrednost, se izkaže, da imajo trije računalniki enako zmogljivost, kadar normaliziramo na X, in tudi enako zmogljivost, če normaliziramo na Y. Ti rezultati so mnogo bolj v skladu z našimi pričakovanji.

2.7 Ocenjevanje zmogljivosti računalnikov (7/7)*

Zamislimo si, da se v nekem programu izvede dva milijona ukazov na procesorju s frekvenco ure 400 MHz. Program je sestavljen iz štirih glavnih tipov ukazov. Ukazna mešanica in števila urinih ciklov na ukaz (CPI) za vsak tip ukazov so podani v spodnji tabeli:

Tip ukaza	**CPI**	**Ukazna mešanica (%)**
Aritmetični in logični	1	60
Dostop do predpomnilnika (cache hit)	2	18
Vejanje	4	12
Dostop do pomnilnika (cache miss)	8	10

Podobno kot v nalogi **2.1** lahko tudi za izvajanje tega programa hitro izračunamo, da je CPI = 2,24 in zmogljivost v MIPS = 178. Zdaj predpostavimo, da je lahko program izvajan v osmih sočasnih opravilih ali nitih, kjer se v vsakem opravilu izvede približno enako število ukazov. Izvajanje poteka na osemjedrnem sistemu, kjer ima vsako jedro (procesor) enako zmogljivost kot en sam procesor, ki je bil uporabljen sprva. Koordinacija in sinhronizacija med opravili privedeta do dodatnih 25.000 izvedenih ukazov za vsako opravilo. Predpostavite enako ukazno množico za vsako opravilo kot v primeru enega samega procesorja, povečajte pa CPI za dostope do pomnilnika ob zgrešitvi predpomnilnika na 12 urinih ciklov zaradi tekme za pomnilnik [1].

a) Določite povprečni CPI.

b) Določite ustrezno zmogljivost v MIPS.

c) Izračunajte pohitritev.

d) Primerjajte dejansko pohitritev s teoretično pohitritvijo, določeno z Amdahlovim zakonom.

Rešitev:

a) Ob predpostavki enake ukazne mešanice to pomeni, da se dodatni ukazi za vsako opravilo razporedijo sorazmerno med posamezne tipe ukazov. Imamo torej naslednjo tabelo:

Tip ukaza	**CPI**	**Ukazna mešanica (%)**
Aritmetični in logični	1	60
Dostop do predpomnilnika (cache hit)	2	18
Vejanje	4	12
Dostop do pomnilnika (cache miss)	12	10

$CPI = 0{,}6 + 2 \cdot 0{,}18 + 4 \cdot 0{,}12 + 12 \cdot 0{,}1 = 2{,}64$

Število CPI se je povečalo zaradi povečanega časa za dostop do pomnilnika.

b)

$$\text{zmogljivost v MIPS} = \frac{400}{2{,}64} \approx 152$$

Zmogljivost v MIPS je zaradi povečanega CPI ustrezno padla.

c) Pohitritev je razmerje časov izvajanja. Za primer z enim samim procesorjem je čas izvajanja enak

$$T_1 = \frac{I_C}{(\text{zmogljivost v MIPS}) \times 10^6} = \frac{2 \cdot 10^6}{178 \cdot 10^6} = 11\,\text{ms}.$$

V primeru z osmimi procesorji vsak procesor izvaja 1/8 od dveh milijonov ukazov, poleg tega pa še 25.000 režijskih ukazov. V primeru z osmimi paralelnimi procesorji je čas izvajanja za vsakega izmed osmih procesorjev enak

$$T_8 = \frac{\frac{2 \cdot 10^6}{8} + 0{,}025 \cdot 10^6}{152 \cdot 10^6} = 1{,}8\,\text{ms}.$$

Pohitritev S znaša torej

$$S = \frac{\text{čas za izvedbo programa na enem samem procesorju}}{\text{čas za izvedbo programa na 8 paralelnih procesorjih}} = \frac{11}{1{,}8} = 6{,}11.$$

d) Odgovor na to vprašanje je odvisen od tega, kako razlagamo Amdahlov zakon. V sistemu z več procesorji sta dve neučinkovitosti. Prvič, za koordinacijo med nitmi so potrebni dodatni ukazi. Drugič, obstaja tekmovanje za dostop do pomnilnika. Glede na to, kako je problem postavljen, noben del kode ni sam po sebi serijski. Vsa koda se da paralelizirati, vendar z dodatno režijo za razporejanja. Lahko bi trdili, da konflikt pri dostopu do pomnilnika pomeni, da do neke mere ukazov za dostop do pomnilnika ni mogoče paralelizirati. Ni pa jasno, kako ovrednotiti ta učinek v Amdahlovi enačbi. Če predpostavimo, da je del kode, ki ga je mogoče paralelizirati, $f = 1$, potem po Amdahlovem zakonu za naš primer dobimo

$$S = \frac{1}{1 - 1 + \frac{1}{8}} = 8.$$

Tako je dejanska pohitritev zgolj okrog $\frac{6{,}11}{8} = 76$ % teoretične pohitritve.

2.8 Amdahlov zakon (1/4)

Na kolikojedrnem procesorju bi morali izvajati program, v katerem se dâ 90 % kode poljubno paralelizirati, da bi dobili faktor pohitritve 5 v primerjavi z izvajanjem istega programa na enojedrnem procesorju?

Rešitev:
Iz splošne enačbe Amdahlovega zakona izrazimo N ter v izraz vstavimo vrednosti $f = 0{,}9$ in $S = 5$.

$$S = \frac{1}{1 - f + \frac{f}{N}}$$

$$S(1 - f + \frac{f}{N}) = 1$$

$$1 - f + \frac{f}{N} = \frac{1}{S}$$

$$\frac{f}{N} = \frac{1}{S} - 1 + f$$

$$N = \frac{f}{\frac{1}{S} - 1 + f} = \frac{0{,}9}{\frac{1}{5} - 1 + 0{,}9} = \frac{0{,}9}{0{,}1} = 9$$

Program bi morali izvajati na devetjedrnem procesorju, da bi pri $f = 0{,}9$ dobili petkratno pohitritev v primerjavi z izvajanjem istega programa na enojedrnem procesorju.

2.9 Amdahlov zakon (2/4)

Najmanj na kolikojedrnem procesorju bi morali izvajati program, v katerem se dâ 80 % kode poljubno paralelizirati, da bi dobili dvakrat večjo pohitritev kot pri dvojedrnem procesorju?

Rešitev:
Najprej zapišemo dve enačbi Amdahlovega zakona, eno za primer z N procesorji in drugo za primer z dvema procesorjema. Nato izrazimo razmerje $\frac{S_N}{S_2}$, ki ga izenačimo z vrednostjo 2. Nazadnje iz dobljene enačbe izrazimo N in ga izračunamo.

$$S_N = \frac{1}{1 - f + \frac{f}{N}}, \quad S_2 = \frac{1}{1 - f + \frac{f}{2}}$$

$$\frac{S_N}{S_2} = \frac{1 - f + \frac{f}{2}}{1 - f + \frac{f}{N}} = 2$$

$$2 - 2f + \frac{2f}{N} = 1 - f + \frac{f}{2}$$

$$\frac{2f}{N} = 1 - f + \frac{f}{2} - 2 + 2f = \frac{3f}{2} - 1$$

$$N = \frac{2f}{\frac{3f}{2} - 1} = \frac{2 \cdot 0{,}8}{\frac{3 \cdot 0{,}8}{2} - 1} = \frac{1{,}6}{1{,}2 - 1} = 8$$

Da bi pri $f = 0{,}8$ dobili dvakrat večjo pohitritev kot pri dvojedrnem procesorju, bi morali program izvajati na osemjedrnem procesorju.

2.10 Amdahlov zakon (3/4)*

Vaše podjetje je pravkar kupilo novi Intel Core i5 dvojedrni procesor in dobili ste nalogo optimizirati vašo programsko opremo za ta procesor. Na tem dvojedrnem procesorju boste izvajali dve aplikaciji, vendar njuni zahtevi po virih nista enaki. Prva aplikacija zahteva 80 % virov, druga pa samo 20 % virov. Predpostavite, da kadar paralelizirate del programa (to pomeni, da ga izvajate na dveh jedrih), je pohitritev za ta del enaka 2 [2].

a) Če je mogoče 40 % prve aplikacije paralelizirati, kolikšno pohitritev bi dosegli, če bi izvajali samo njo?

b) Če je mogoče 99 % druge aplikacije paralelizirati, kolikšno pohitritev bi dosegli, če bi izvajali samo njo?

c) Če je mogoče 40 % prve aplikacije paralelizirati, kolikšno celotno pohitritev sistema bi dosegli, če jo paralelizirate?

d) Če je mogoče 99 % druge aplikacije paralelizirati, kolikšno celotno pohitritev sistema bi dosegli, če jo paralelizirate?

Rešitev:

a) Pohitritev S izolirano izvajane prve aplikacije je:

$$S = \frac{1}{0{,}6 + \frac{0{,}4}{2}} = 1{,}25$$

b) Pohitritev S izolirano izvajane druge aplikacije je:

$$S = \frac{1}{0{,}01 + \frac{0{,}99}{2}} = 1{,}98$$

c) Pohitritev S celotnega sistema (hkrati se izvajata obe aplikaciji) s paraleliziranim izvajanjem prve aplikacije je:

$$S = \frac{1}{0{,}2 + 0{,}8 \cdot (0{,}6 + \frac{0{,}4}{2})} = \frac{1}{0{,}2 + 0{,}8 \cdot 0{,}8} = 1{,}19$$

d) Pohitritev S celotnega sistema (hkrati se izvajata obe aplikaciji) s paraleliziranim izvajanjem druge aplikacije je:

$$S = \frac{1}{0{,}8 + 0{,}2 \cdot (0{,}01 + \frac{0{,}99}{2})} = \frac{1}{0{,}8 + 0{,}2 \cdot 0{,}505} = 1{,}11$$

2.11 Amdahlov zakon (4/4)*

Ko paraleliziramo aplikacijo, jo pohitrimo z večanjem števila procesorjev. To je omejeno z dvojim: deležem oziroma odstotkom aplikacije, ki se lahko paralelizira, in ceno komunikacije. Amdahlov zakon upošteva prvo omejitev, druge pa ne [2].

a) Kolikšna je pohitritev z N procesorji, če je 80 % aplikacije mogoče paralelizirati in zanemarimo ceno komunikacije?

b) Kolikšna je pohitritev z osmimi procesorji, če je za vsak dodani procesor režija za komunikacijo 0,5 % originalnega časa izvajanja?

c) Kolikšna je pohitritev z osmimi procesorji, če se vsakokrat, ko se število procesorjev podvoji, režija za komunikacijo poveča za 0,5 % originalnega časa izvajanja?

d) Kolikšna je pohitritev z N procesorji, če se vsakokrat, ko se število procesorjev podvoji, režija za komunikacijo poveča za 0,5 % originalnega časa izvajanja?

e) Napišite splošno enačbo, ki reši naslednje vprašanje. Kolikšno je število procesorjev z največjo pohitritvijo v aplikaciji, v kateri se dâ P % originalnega izvajalnega časa paralelizirati, in se vsakokrat, ko se število procesorjev podvoji, režija za komunikacijo poveča za 0,5 % originalnega časa izvajanja?

Rešitev:

a) Pohitritev S z N procesorji je:

$$S = \frac{1}{0{,}2 + \frac{0{,}8}{N}}$$

b) Pohitritev S z osmimi procesorji je:

$$S = \frac{1}{0{,}2 + 8 \cdot 0{,}005 + \frac{0{,}8}{8}} = \frac{1}{0{,}2 + 0{,}04 + 0{,}1} = 2{,}94$$

c) Število procesorjev se podvoji trikrat: ko dodamo drugega, četrtega in osmega. Režija za komunikacijo je zato enaka $3 \cdot 0{,}005$. Pohitritev S z osmimi procesorji tako znaša:

$$S = \frac{1}{0{,}2 + 3 \cdot 0{,}005 + \frac{0{,}8}{8}} = \frac{1}{0{,}2 + 0{,}015 + 0{,}1} = 3{,}17$$

d) Pri N procesorjih pride do podvojitve števila procesorjev takrat, ko je $N > 1$ in $\log_2 N$ potenca števila 2. Pohitritev S z osmimi procesorji je:

$$S = \frac{1}{0{,}2 + 0{,}005 \cdot \log_2 N + \frac{0{,}8}{N}}$$

e) V aplikaciji se dâ P % originalnega izvajalnega časa paralelizirati, ostane torej $(1 - P)$ % originalnega izvajalnega časa, ki se ga ne dâ paralelizirati. Režijo za komunikacijo izrazimo tako kot v točki **d)**. Do števila procesorjev, pri katerem

imamo največjo pohitritev, pridemo tako, da izraz za pohitritev odvajamo po N in odvod izenačimo z 0.

$$\frac{d}{dN}S = \frac{d}{dN}\left(\frac{1}{(1-P) + 0{,}005 \cdot \log_2 N + \frac{P}{N}}\right) = 0$$

Iz dobljenega izraza izračunamo optimalni N, pri katerem je pohitritev največja.

3 Komunikacija med glavnimi komponentami računalnika

3.1 Obseg pomnilniškega naslovnega prostora

Nek 16-bitni mikroprocesor ima 23 naslovnih linij. Izračunajte njegov pomnilniški naslovni prostor v megabajtih (MB). Koliko 16-bitnih besed je mogoče shraniti v njegov pomnilnik [3]?

Rešitev:
Pomnilniški naslovni prostor je enak $2^{23} = 8.388.608$ bajtov $= 8\,\text{MB}$. Za hranjenje 16-bitne besede potrebujemo dvobajtne lokacije. Če je mikroprocesor opremljen s celotnim pomnilnikom, ki ga lahko naslavlja, lahko vanj shranimo $\frac{8.388.608}{2} = 4.194.304$ 16-bitnih besed.

3.2 Obseg V/I naslovnega prostora

Naslovno vodilo 16-bitnega mikroprocesorja je sestavljeno iz 23 linij. Za naslavljanje V/I lokacij uporablja samo 16 spodnjih naslovnih linij. Izračunajte obseg V/I naslovnega prostora te CPE. Kateri je najvišji V/I naslov [3]?

Rešitev:
Ker je za V/I operacije uporabljenih le 16 naslovnih linij, obsega V/I naslovni prostor $2^{16} = 65.536$ bajtov (B) oziroma 64 kilobajtov (KB). Najvišji V/I naslov, ki ga CPE lahko doseže, je $65.536 - 1 = 65.535$. Najnižji V/I naslov je, seveda, 0.

3.3 Vpliv širine vodil na hitrost sistema

Zamislite si hipotetični 32-bitni mikroprocesor, ki ima 32-bitne ukaze, sestavljene iz dveh polj: prvi bajt vsebuje operacijsko kodo, preostali pa sprotni operand ali naslov operanda [1].

a) Koliko pomnilnika je mogoče neposredno nasloviti (v bajtih)?

b) Komentirajte vpliv na hitrost sistema, če ima mikroprocesorsko vodilo

1. 32-bitno lokalno naslovno vodilo in 16-bitno lokalno podatkovno vodilo ali
2. 16-bitno lokalno naslovno vodilo in 16-bitno lokalno podatkovno vodilo.

c) Koliko bitov je potrebnih za programski števnik in ukazni register?

Rešitev:

a) Neposredno je mogoče nasloviti $2^{24} = 16\,\text{MB}$ pomnilnika.

b) 1. Če je lokalno naslovno vodilo 32-bitno, se lahko celoten naslov prenese k pomnilniku naenkrat. Ker pa je podatkovno vodilo samo 16-bitno, bosta potrebna dva cikla za včitanje 32-bitnega ukaza ali operanda.

2. 16 bitov naslova, postavljenega na naslovno vodilo, ne more doseči celotnega pomnilnika. Zato potrebujemo bolj obsežen vmesnik pomnilnika, ki bo zadržal prvi del naslova in nato še drugega (mikroprocesor mora napraviti dva dostopa). Za 32-bitni naslov lahko predpostavimo, da bomo s prvo polovico naslova dekodirali „vrstico“ v pomnilniku, kasneje pa z drugo polovico naslova „stolpec“ v pomnilniku. Poleg „dvokoračnega“ delovanja naslovov bo moral mikroprocesor opraviti dva cikla za včitanje 32-bitnega ukaza ali operanda.

c) Programski števnik mora biti vsaj 24-bitni. Navadno ima 32-bitni mikroprocesor 32-bitno zunanje naslovno vodilo in 32-bitni programski števnik, razen če se v čipu uporabljajo segmentni registri, ki lahko delajo z manjšim programskim števnikom. Če naj ukazni register vsebuje celoten ukaz, mora biti dolg 32 bitov. Če bo vseboval samo operacijsko kodo, mora biti dolg 8 bitov.

3.4 Pomnilniški in V/I naslovni prostor

Zamislite si hipotetični mikroprocesor, ki generira 16-bitne naslove (predpostavite, da so programski števnik in naslovni registri 16-bitni) in ima 16-bitno podatkovno vodilo [1].

a) Koliko znaša maksimalna velikost pomnilniškega naslovnega prostora, ki ga lahko procesor neposredno doseže, če je povezan s „16-bitnim pomnilnikom“?

b) Koliko znaša maksimalna velikost pomnilniškega naslovnega prostora, ki ga lahko procesor neposredno doseže, če je povezan z „8-bitnim pomnilnikom“?

c) Katera arhitekturna značilnost bi mikroprocesorju omogočala dostop do ločenega V/I prostora?

d) Koliko 8-bitnih V/I vrat lahko mikroprocesor podpira, če lahko vhodni in izhodni ukaz specificirata 8-bitno številko V/I vrat? Koliko pa lahko podpira 16-bitnih V/I vrat?

Rešitev:

a) Mikroprocesor lahko neposredno doseže $2^{16} = 64\,\text{KB}$ pomnilnika. Med dostopom lahko prenese 16-bitno besedo ali en bajt.

b) Mikroprocesor lahko neposredno doseže $2^{16} = 64\,\text{KB}$ pomnilnika. Med dostopom lahko prenese en bajt.

c) Mikroprocesor bi potreboval posebna ukaza za branje iz V/I lokacije in za pisanje na V/I lokacijo, katerih izvajanje bi generiralo posebni krmilni signal (npr. M/IO*), ki bi označeval, ali želi mikroprocesor dostopiti do pomnilnika ali V/I lokacije. Mikroprocesor bi potreboval najmanj en dodaten izhodni priključek.

d) Mikroprocesor lahko podpira $2^8 = 256$ vhodnih in $2^8 = 256$ izhodnih 8-bitnih vrat ter prav toliko 16-bitnih vhodnih in izhodnih vrat.

3.5 V/I modul za nadzor teleprinterja

Zamislite si računalniški sistem, ki vsebuje V/I modul za nadzor teleprinterja s preprosto tipkovnico in tiskalnikom. V V/I modulu so naslednji registri, ki so neposredno povezani na sistemsko vodilo:

INPR: Input Register, 8 bitov,
OUTR: Output Register, 8 bitov,
FGI: Input Flag, 1 bit,
FGO: Output Flag, 1 bit,
IEN: Interrupt Enable, 1 bit.

V/I modul nadzira pritiske na tipke tipkovnice in izpise na tiskalnik teleprinterja. Teleprinter je sposoben zakodirati alfanumerične simbole v 8-bitno besedo in dekodirati 8-bitno besedo v alfanumerični simbol [1].

a) Opišite, kako procesor z uporabo prvih štirih navedenih registrov upravlja V/I prenose na teleprinterju?

b) Opišite, kako je mogoče funkcijo izvajati učinkoviteje z uporabo IEN.

Rešitev:

a) Vnos s tipkovnice teleprinterja se shrani v register INPR. INPR bo sprejel podatek iz teleprinterja, ko je zastavica FGI=0. Ko podatek prispe, se shrani v INPR, FGI pa se postavi na 1. CPE periodično preverja FGI. Če je FGI =1, CPE prenese vsebino registra INPR v akumulator AC in zbriše FGI na 0. Ko ima CPE podatek za pošiljanje na teleprinter, preveri zastavico FGO. Če je FGO = 0, mora CPE počakati. Če je FGO = 1, CPE prenese vsebino iz akumulatorja AC v register OUTR in zbriše zastavico FGO na 0. Teleprinter postavi zastavico FGI na 1, ko je beseda natisnjena.

b) Proces, opisan v točki **a)**, je zelo potraten. CPE, ki je veliko hitrejša od teleprinterja, mora vedno znova preverjati zastavici FGI in FGO. Če se uporabljajo prekinitve, lahko teleprinter sproži prekinitev k CPE, kadarkoli je pripravljen sprejeti ali poslati podatek.

3.6 Trajanje strojnega cikla mikroprocesorja

Mikroprocesor je poganjan z uro frekvence 5 MHz. Kako dolg je urin cikel? Koliko časa traja nek tip strojnega cikla, sestavljen iz treh urinih ciklov [3]?

Rešitev:
Perioda ure T je enaka $\frac{1}{f}$, kjer je f frekvenca ure. Iz tega sledi, da je trajanje urinega cikla $\frac{1}{5 \cdot 10^6} = 0{,}2 \cdot 10^{-6}$ s $= 200$ ns. Trajanje strojnega cikla s tremi urinimi cikli je $3T = 3 \cdot 200\,\text{ns} = 600\,\text{ns}$.

3.7 Podvojitev širine vodila proti podvojitvi frekvence ure

Zamislite si 32-bitni mikroprocesor s 16-bitnim zunanjim podatkovnim vodilom, ki ga poganja ura s frekvenco 8 MHz. Predpostavite, da cikel na vodilu tega mikroprocesorja znaša najmanj štiri urine cikle. Koliko znaša najvišja hitrost prenosa

podatkov prek vodila v bajtih na sekundo, ki jih zmore mikroprocesor? Ali bi bilo za povečanje zmogljivosti bolje podvojiti širino zunanjega podatkovnega vodila na 32 bitov ali podvojiti frekvenco ure mikroprocesorja? Komentirajte, kakšne spremembe bi zahtevala ena ali druga odločitev. *Namig:* Določite število bajtov, ki se lahko prenesejo v enem ciklu na vodilu [1].

Rešitev:
En urin cikel (perioda ure) traja $\tau = 1/(8\,\mathrm{MHz}) = 125\,\mathrm{ns}$. Najkrajši možni cikel na vodilu traja $4 \cdot 125\,\mathrm{ns} = 500\,\mathrm{ns}$. Ker je vodilo 16-bitno, se v 500 ns prek vodila preneseta 2 bajta. V 1 sekundi se prek vodila prenese $(2\ \mathrm{B})/(500\,\mathrm{ns}) = 4\,\mathrm{MB}$ podatkov. Najvišja hitrost prenosa podatkov je tako enaka 4 MB/s. Tako podvojitev širine zunanjega podatkovnega vodila kot podvojitev frekvence ure bi podvojila najvišjo dosegljivo hitrost prenosa podatkov prek vodila. Podvojitev frekvence ure bi lahko pomenila potrebo po uporabi druge polprevodniške tehnologije pri izdelavi čipa (predpostavimo, da bi vsak ukaz trajal enako število urinih ciklov kot v čipu z originalno tehnologijo). Podvojitev širine zunanjega podatkovnega vodila bi pomenila podvojitev števila gonilnikov in zadrževalnikov podatkovnega vodila v samem čipu in spremembe pri krmilni logiki vodila. V prvem primeru bi se morala tudi hitrost pomnilnika (približno) podvojiti, da pomnilnik ne bi upočasnjeval mikroprocesorja. V drugem primeru bi se morala podvojiti dolžina besede pomnilnika, da bi pomnilnik lahko sprejemal in pošiljal 32-bitne besede.

3.8 Propustnost vodila za 8- in 16-bitni mikroprocesor

Imamo dva mikroprocesorja, ki imata 8- in 16-bitno zunanje podatkovno vodilo. Sicer sta obe CPE identični in imata enako dolge cikle na vodilu. Predpostavite, da so vsi ukazi in operandi dolgi dva bajta. Za kakšen faktor se razlikujeta njuni propustnosti vodila? Kaj pa, če je 50 % operandov in ukazov enobajtnih [3]?

Rešitev:
V enem ciklu na vodilu prenese 8-bitni mikroprocesor samo en bajt, 16-bitni pa dva. Torej ima 16-bitni mikroprocesor dvakrat večjo propustnost vodila. Zamislimo si 100 ukazov (in/ali operandov), od katerih jih je 50 dolgih en bajt in 50 dva bajta. 8-bitni mikroprocesor zahteva za njihov prenos $2 \cdot 50 + 50 = 150$ ciklov na vodilu. 16-bitni mikroprocesor zahteva $50 + 50 = 100$ ciklov na vodilu. Njuni propustnosti vodila se zdaj razlikujeta samo za faktor 1,5.

3.9 Cikli na vodilu za 16- in 32-bitni mikroprocesor

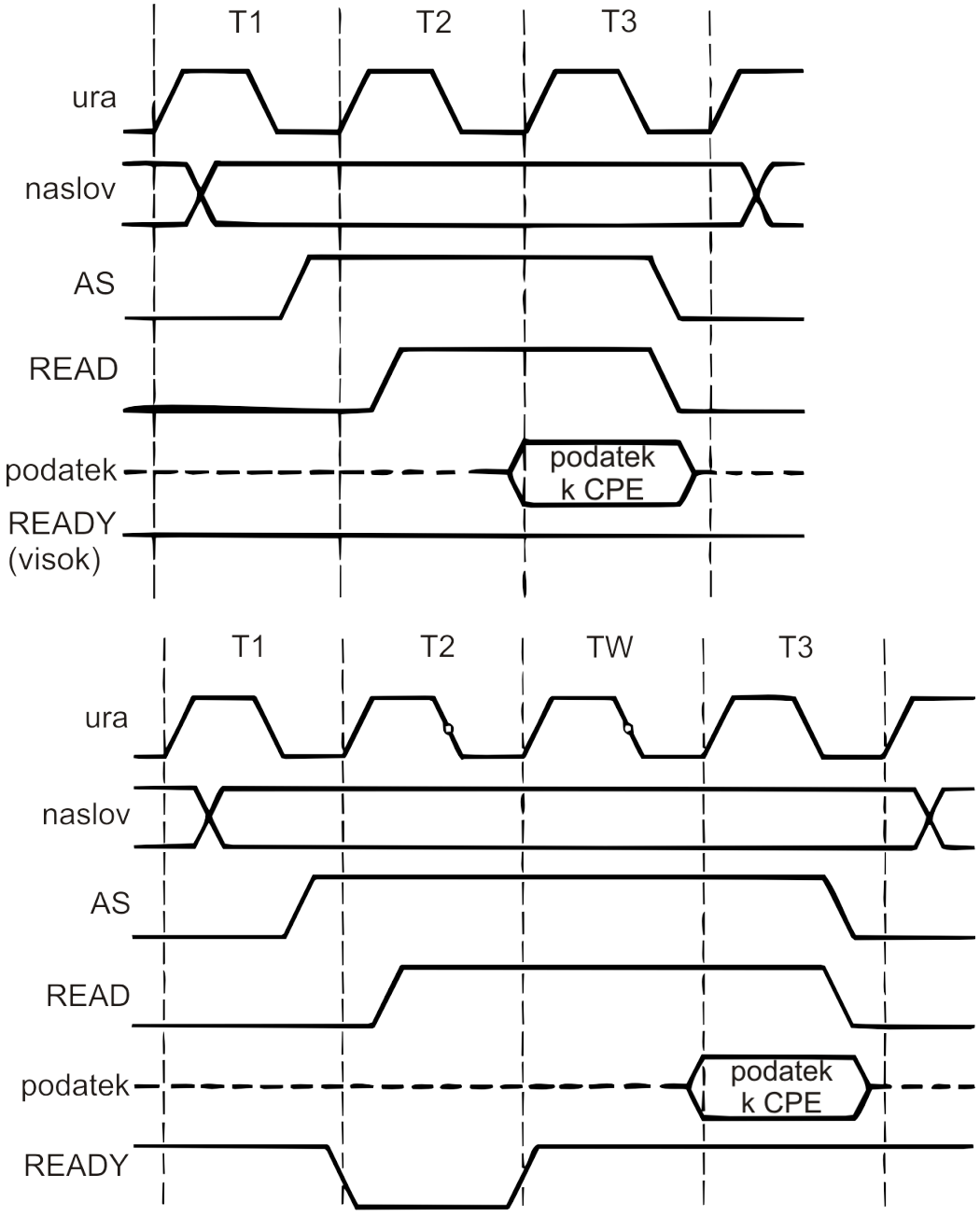
Zamislite si 32-bitni mikroprocesor, katerega cikel na vodilu traja enako dolgo kot cikel na vodilu 16-bitnega mikroprocesorja. Predpostavite, da je v povprečju 20 % operandov in ukazov dolgih 32 bitov, 40 % jih je dolgih 16 bitov in 40 % jih je dolgih le 8 bitov. Izračunajte doseženo izboljšanje, če včitujete ukaze in operande z 32-bitnim mikroprocesorjem [1].

Rešitev:
Zamislimo si mešanico 100 ukazov in operandov. V povprečju je sestavljena iz 20 32-bitnih besed, 40 16-bitnih besed in 40 bajtov. Število potrebnih ciklov na vodilu za 16-bitni mikroprocesor je $2 \cdot 20 + 40 + 40 = 120$. Za 32-bitni mikroprocesor je

potrebnih $20 + 40 + 40 = 100$ ciklov na vodilu. Doseženo izboljšanje v odstotku zmanjšanja potrebnih ciklov na vodilu pri uporabi 32-bitnega procesorja namesto 16-bitnika znaša $(120 - 100)/120 = 17$ %.

3.10 Pomnilniški bralni cikel na sinhronem vodilu (1/2)

Nek mikroprocesor ima sinhrono vodilo s takšnim časovnim potekom pomnilniškega bralnega cikla, kot je prikazan na spodnji sliki.



Pri operaciji branja na sinhronem vodilu mora pomnilniški modul postaviti podatek na vodilo dovolj časa pred negativno fronto signala READ, da se signali na podatkovnem vodilu že umirijo. Ob negativni fronti ure v T2 CPE testira vhodni signal READY, da se prepriča, ali bo podatek iz pomnilnika na voljo pred koncem T3. Če je vrednost signala READY takrat visoka, je to zagotovilo, da bo podatek iz

pomnilnika na voljo pravočasno in CPE lahko vstopi v stanje T3. Če je vrednost signala READY ob testiranju nizka, CPE ne sme vstopiti v stanje T3, pač pa mora strojni cikel podaljšati z vstavitvijo čakalnega stanja TW. Če še to ne zadostuje, mora CPE vriniti dodatna čakalna stanja. Predpostavite, da ima mikroprocesor uro frekvence 10 MHz in da ima signal READ negativno fronto v sredini druge polovice periode T3 [3].

a) Izračunajte dolžino pomnilniškega bralnega cikla.

b) Koliko časa po začetku cikla CPE preverja linijo READY?

c) Kdaj najpozneje mora biti pomnilniški podatek postavljen na vodilo? Upoštevajte 20 ns za umiritev podatkovnih linij. Ignorirajte čase vzpona in padca signalov.

Rešitev:

a) Pri frekvenci ure 10 MHz znaša perioda ure 100 ns. Dolžina pomnilniškega bralnega cikla je $3 \cdot 100 = 300$ ns.

b) CPE testira linijo READY v sredini T2, ki je 150 ns oddaljena od začetka strojnega cikla.

c) Negativna fronta signala READ nastopi $100 + 100 + 75 = 275$ ns po začetku strojnega cikla (v sredini druge polovice periode T3). Da bi se signali na podatkovnem vodilu lahko umirili, mora pomnilnik postaviti podatek na vodilo vsaj $275 - 20 = 255$ ns po začetku cikla ali z drugimi besedami najpozneje 55 ns po prvi fronti T3.

3.11 Pomnilniški bralni cikel na sinhronem vodilu (2/2)

Zamislite si mikroprocesor, ki ima takšen potek pomnilniškega bralnega cikla, kot je prikazan na sliki pri nalogi **3.10**. Po nekaj analizah načrtovalec ugotovi, da pomnilnik zamuja s pravočasnim zagotavljanjem prebranega podatka za okrog 180 ns [3].

a) Koliko čakalnih stanj (urinih ciklov) se mora vstaviti za pravilno delovanje sistema, če je frekvenca ure 8 MHz?

b) Za vrivanje čakalnih stanj se uporablja vhodna statusna linija READY mikroprocesorja. Ko procesor izda ukaz READ za branje, mora s poskusom branja podatka počakati vse dotlej, dokler se ne aktivira (postavi) linija READY. Znotraj katerega časovnega intervala moramo držati linijo READY v nizkem stanju, da prisilimo procesor k vstavitvi zahtevanega števila čakalnih stanj?

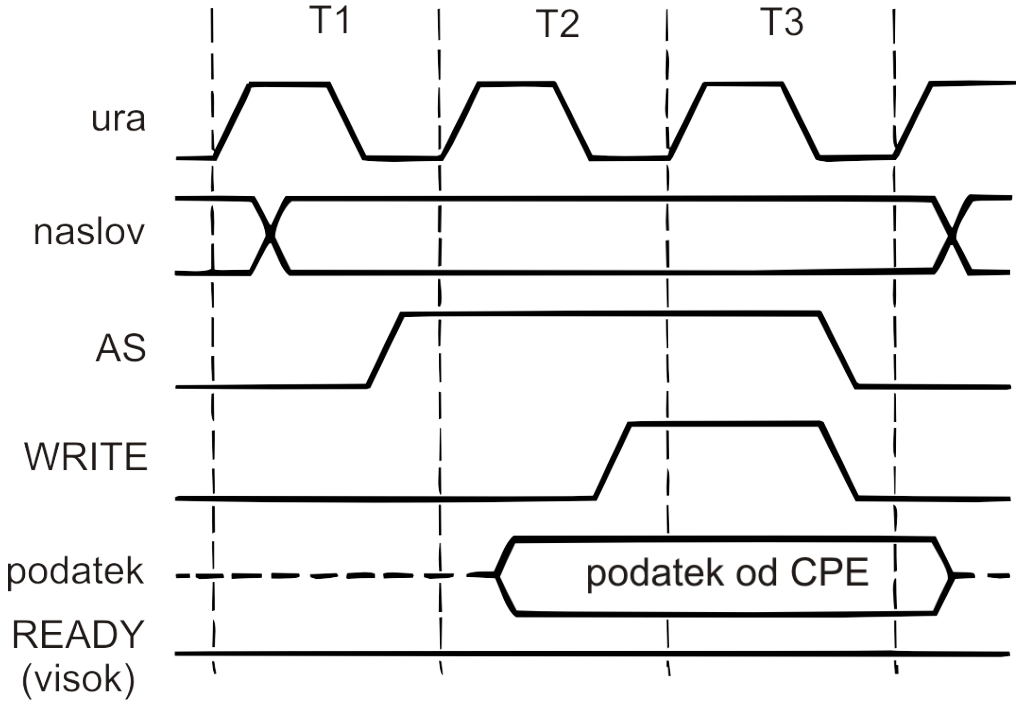
Rešitev:

a) Perioda ure je $1/(8\,\text{MHz}) = 125$ ns, torej vsako čakalno stanje podaljša pomnilniški bralni cikel za 125 ns. Ker moramo podaljšati pomnilniški bralni cikel za 180 ns, moramo vstaviti dva urina cikla. To dejansko podaljša cikel branja za 250 ns in ostane rezerva $250 - 180 = 70$ ns.

b) Da bi prisilili CPE k vstavitvi dveh čakalnih stanj, moramo spraviti READY v nizko stanje pred negativno fronto T2. Od tam naprej moramo držati READY v nizkem stanju do negativne fronte prvega čakalnega stanja. Negativna fronta T2 se pojavi $125 + 62{,}5 = 187{,}5$ ns od začetka strojnega cikla. Negativna fronta prvega stanja TW se pojavi $187{,}5 + 125 = 312{,}5$ ns od začetka strojnega cikla. READY mora zavzeti nizko stanje malenkost pred točko 187,5 ns (set-up time) in na njem ostati malenkost za točko 312,5 ns (hold time).

3.12 Pomnilniški pisalni cikel na sinhronem vodilu

Nek mikroprocesor ima takšen potek pisanja v pomnilnik, kot je prikazan na spodnji sliki. Njegov proizvajalec specificira, da je širina signala WRITE določena z izrazom $T - 50$, kjer je T perioda ure v ns [3].



a) Kolikšno širino signala WRITE lahko pričakujemo, če ima ura frekvenco 5 MHz?

b) V priročniku za mikroprocesor je podano, da ostanejo podatki veljavni še 20 ns po negativni fronti signala WRITE. Koliko časa skupno je na vodilu veljaven podatek?

c) Koliko čakalnih stanj moramo vstaviti, če pomnilnik zahteva veljavnost podatka na vodilu vsaj 190 ns?

Rešitev:

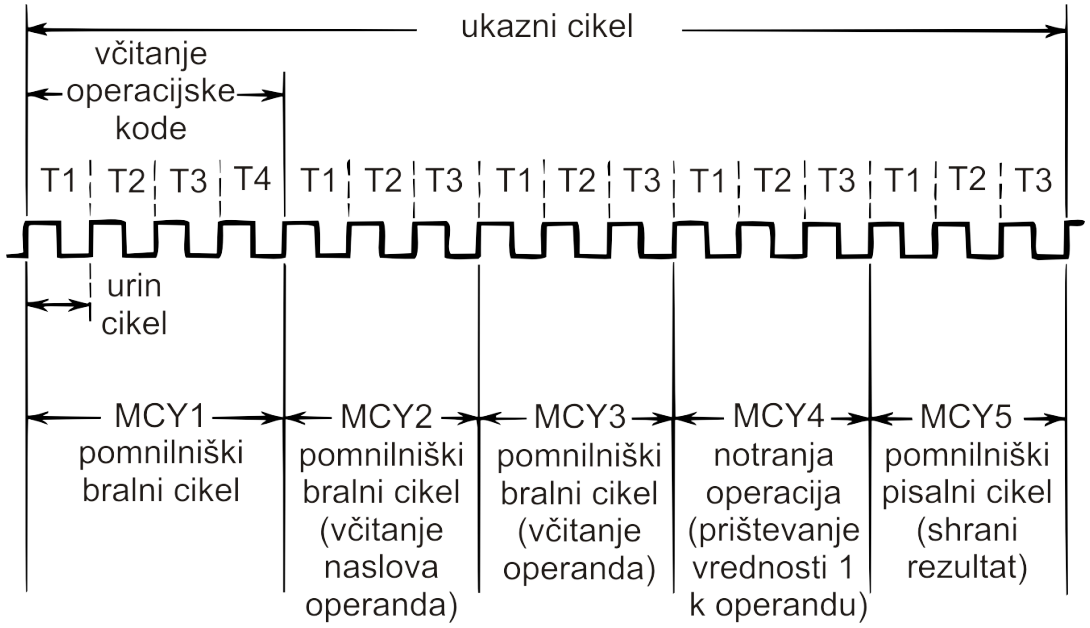
a) Ura frekvence 5 MHz ustreza periodi 200 ns. Iz tega sledi, da je pričakovana širina signala WRITE $200 - 50 = 150$ ns.

b) Podatek ostane na vodilu veljaven $150 + 20 = 170$ ns.

c) Če pomnilnik zahteva veljavnost podatka na vodilu najmanj 190 ns, moramo vstaviti eno čakalno stanje. To poveča interval prisotnosti veljavnega podatka na vodilu na $170 + 200 = 370$ ns.

3.13 Vrivanje čakalnih stanj v ukazni cikel ukaza INC

Nek mikroprocesor ima ukaz (INC) za neposredno inkrementiranje pomnilniške lokacije, ki prišteje 1 k vrednosti v pomnilniški lokaciji. Ukaz ima pet faz: včitanje operacijske kode (štirje urini cikli na vodilu), včitanje naslova operanda (trije cikli), včitanje operanda (trije cikli), prištevanje vrednosti 1 k operandu (trije cikli) in shranitev rezultata (trije cikli). Časovni potek ukaznega (inštrukcijskega) cikla je prikazan spodaj [1].



a) Za koliko odstotkov se bo podaljšal ukazni cikel, če moramo v vsak pomnilniški bralni ali pomnilniški pisalni cikel na vodilu vstaviti dve čakalni stanji?

b) Izračun ponovite ob predpostavki, da operacija inkrementiranja namesto treh ciklov traja 13 ciklov.

Rešitev:

a) Brez čakalnih stanj traja ukazni cikel 16 urinih ciklov. Ukaz zahteva štiri dostope do pomnilnika, kar pomeni osem čakalnih stanj. Ukazni cikel skupaj s čakalnimi stanji traja 24 urinih ciklov, torej se je podaljšal za 50 %.

b) V tem primeru bi ukazni cikel trajal 26 urinih ciklov brez čakalnih stanj in 34 ciklov s čakalnimi stanji, kar pomeni povečanje za 31 %.

3.14 Čas izvajanja ukaza INC

Za mikroprocesor iz primera **3.13** predpostavite frekvenco ure 8 MHz. Koliko traja včitavanje in izvajanje inštrukcije INC? Koliko je največje število inštrukcij INC, ki jih CPE lahko izvede na sekundo [3]?

Rešitev:
Celotna dolžina ukaznega cikla je 16 urinih ciklov. Urin cikel traja $\frac{1}{8 \cdot 10^6} = 125\,\text{ns}$. Tako traja celoten ukazni cikel $16 \cdot 125 = 2.000\,\text{ns} = 2\,\mu\text{s}$. Naša hipotetična CPE lahko izvede do $\frac{1}{2 \cdot 10^{-6}} = 500.000$ ukazov INC na sekundo.

3.15 Vpliv vrivanja čakalnih stanj na dolžino ukaznega cikla

Spomnimo se spet ukaznega cikla ukaza INC iz primera **3.13**. Za koliko odstotkov se poveča dolžina cikla, če moramo v vsak cikel na vodilu vstaviti dvoje čakalnih stanj? Ponovite izračun ob predpostavki, da strojni cikel za notranje operacije traja 51 urinih ciklov [3].

Rešitev:
Normalno traja ukazni cikel 16 stanj T. Dodatek dveh čakalnih stanj v bralni in pisalni strojni cikel poveča dolžino ukaznega cikla za osem stanj T, kar znaša 50 %. Če predpostavimo 51 stanj T za cikel notranjih operacij, postane normalna dolžina ukaznega cikla 64 stanj T. Zdaj predstavlja osem čakalnih stanj povečanje samo za $\frac{8}{64} = 0{,}125$, to je za 12,5 %.

Ta primer kaže, kako je negativni vpliv dodajanja čakalnih stanj odvisen od dolžine cikla notranjih operacij. Nekateri aritmetični ukazi (množenje, deljenje) imajo dolge cikle z notranjimi operacijami. V aplikacijah z mnogimi računskimi operacijami je vpliv relativno majhen. Po drugi strani pa je lahko v aplikacijah, kjer CPE porabi večino svojega časa za premikanje podatkov, vpliv zelo velik.

3.16 Generiranje prekinitve med izvajanjem ukaza INC

Mikroprocesor pri nalogi **3.13** začne fazo včitavanja operanda ukaza za neposredno inkrementiranje pomnilnika v tistem trenutku, ko tipkovnica aktivira linijo za zahtevo po prekinitvi. Po kolikšnem času procesor vstopi v procesiranje prekinitve? Predpostavite uro frekvence 10 MHz [3].

Rešitev:
V trenutku, ko tipkovnica sproži zahtevo po prekinitvi, potrebuje procesor še devet urinih ciklov do zaključitve izvajanja ukaza. Procesiranje prekinitve se bo torej začelo po preteku $9 \cdot 100 = 900$ ns.

3.17 Prekinitve pri prekinljivih in neprekinljivih ukazih

Nek mikroprocesor ima ukaz za premikanje niza bajtov z enega področja pomnilnika v drugo. Včitavanje in dekodiranje ukaza traja deset urinih ciklov. Nato prenos vsakega bajta traja 15 urinih ciklov. Izračunajte dolžino ukaznega cikla za primer niza dolžine 64 bajtov, če je mikroprocesor poganjan z 10 MHz. Koliko znaša največja možna zakasnitev odziva na prekinitev, če ukaza ni mogoče prekiniti? Ponovite izračun za primer, če se ukaz lahko prekine na začetku vsakega prenosa bajta [3].

Rešitev:
Dolžina urinega cikla za dano frekvenco ure je 100 ns. Zaradi tega je dolžina ukaznega cikla $(10 + 15 \cdot 64) \cdot 100 = 97\,\mu$s. Če ukaza ni mogoče prekiniti, znaša največja možna zakasnitev odziva na prekinitev $97\,\mu$s. Če pa je ukaz mogoče prekiniti tik pred začetkom vsakega prenosa bajta, znaša zgornja zakasnitev le $15 \cdot 100 = 1{,}5\,\mu$s.

3.18 Največja možna zakasnitev potrditve zahteve za vodilo

Najdaljši strojni cikel CPE traja pet urinih ciklov. Določite največjo možno zakasnitev potrditve zahteve za vodilo, če je CPE poganjana z uro frekvence 5 MHz [3].

Rešitev:
Do največje zakasnitve pride, če se zahteva za vodilo pojavi natanko po začetku strojnega cikla dolžine petih urinih ciklov. Ker je perioda ure 200 ns, je zakasnitev enaka $5 \cdot 200 = 1\,\mu$s.

3.19 Hitrost prenosa podatkov mikroprocesorja Intel 8088

Mikroprocesor Intel 8088 ima časovni potek pomnilniškega bralnega cikla, kot je prikazan na sliki pri nalogi **3.10**, vendar pa zahteva štiri urine cikle. Veljaven podatek je na vodilu v časovnem intervalu, ki sega v četrti urin cikel. Predpostavite, da je mikroprocesor poganjan z uro frekvence 8 MHz [3].

a) Koliko znaša najvišja hitrost prenosa podatkov?

b) Ponovite izračun ob predpostavki, da moramo vriniti eno čakalno stanje na vsak preneseni bajt.

Rešitev:

a) Perioda ure je 125 ns. En pomnilniški bralni cikel traja 500 ns = 0,5 μs. Če se cikli na vodilu ponavljajo eden za drugim, lahko dosežemo hitrost prenosa podatkov $\frac{1\,\mathrm{B}}{0{,}5\,\mu\mathrm{s}} = 2\,\mathrm{MB/s}$.

b) Čakalno stanje podaljša cikel branja na vodilu za 125 ns in tako znaša 625 ns = 0,625 μs. Hitrost prenosa podatkov se je zmanjšala na $\frac{1\,\mathrm{B}}{0{,}625\,\mu\mathrm{s}} = 1{,}6\,\mathrm{MB/s}$.

3.20 Zapravljeni cikli pri včitavanju ukazov za Intel 8088

Predpostavite, da mikroprocesor Intel 8088 z ukazno vrsto dolžine štirih bajtov za včitavanje ukazov iz pomnilnika izvaja program, v katerem je verjetnost skoka (rezultat izvedbe ukaza za skok) 10 %. Zaradi enostavnosti še poenostavite, da so vsi ukazi dolgi dva bajta [3].

a) Kolik delež ciklov na vodilu za včitavanje ukazov je lahko zapravljen?

b) Ponovite izračun, če bi bila ukazna vrsta dolga osem bajtov.

Rešitev:

a) Pojav programskega skoka pomeni, da so bili zaman porabljeni (torej zapravljeni) štirje cikli na vodilu (ustrezajo štirim bajtom, ki so že v ukazni vrsti, ko naletimo na ukaz za skok). Za 100 ukazov je število ciklov, ki na vodilu niso porabljeni zaman, v povprečju $90 \cdot 2 = 180$. Zaman porabljenih in torej zavrženih je lahko do $10 \cdot 4 = 40$ ciklov. Delež zavrženih ciklov je torej največ $\frac{40}{180\,+\,40} = 18\,\%$.

b) Če bi bila kapaciteta ukazne vrste dvakrat večja, bi bil delež zavrženih ciklov na vodilu lahko do $\frac{80}{180\,+\,80} = 30\,\%$.

3.21 Liho uvrščene besede pri mikroprocesorju Intel 8086

Intel 8086 je 16-bitni mikroprocesor, medtem ko je mikroprocesor 8088 njegova 8-bitna različica. Mikroprocesor 8086 uporablja 16-bitno vodilo, ki lahko naenkrat prenese dva bajta, če je nižji bajt na sodem naslovu. 8086 omogoča tako sodo kot liho uvrščene besedne operande. Za dostop do liho uvrščene besede sta potrebna dva pomnilniška cikla (vsak je sestavljen iz štirih urinih ciklov) za prenos besede. Zamislite si ukaz mikroprocesorja 8086, ki vključuje dva 16-bitna operanda. Koliko časa je potrebnega, da se včitata oba operanda? Upoštevajte vse možnosti. Predpostavite frekvenco ure 4 MHz in da ni čakalnih stanj [3].

Rešitev:
Urin cikel znaša 0,25 μs, zato traja pomnilniški cikel 1 μs. Če sta oba operanda sodo uvrščena, traja včitanje obeh 2 μs. Če je eden izmed operandov uvrščen liho, drugi pa sodo, traja včitanje obeh 3 μs. Če sta oba operanda uvrščena liho, traja včitanje obeh 4 μs.

4 Predpomnilnik

4.1 Izračun parametrov predpomnilnika (1/8)

Set-asociativni predpomnilnik je sestavljen iz 64 linij, ki so razdeljene v sete po štiri linije. Glavni pomnilnik vsebuje 4K blokov, vsak blok pa ima 128 besed. Prikažite format naslovov glavnega pomnilnika [1].

Rešitev:
Število setov v predpomnilniku $v = \frac{64}{4} = 16 = 2^4$, zato potrebujemo 4 bite za označitev številke seta.

Glavni pomnilnik je sestavljen iz $4\text{K} = 2^{12}$ blokov, zato za označitev številke bloka potrebujemo 12 bitov. Ti biti so sestavljeni iz bitov značke in bitov za izbiro seta. Ker imamo 4 bite za označitev seta, je polje z značko dolgo $12 - 4 = 8$ bitov. Vsak blok je sestavljen iz $128 = 2^7$ besed, zato potrebujemo 7 bitov za označitev besede.

Naslov glavnega pomnilnika:

ZNAČKA	SET	BESEDA
8	4	7

4.2 Izračun parametrov predpomnilnika (2/8)

Dvakrat set-asociativni predpomnilnik ima linije dolge 16 bajtov in skupno velikost 8 KB. Glavni pomnilnik velikosti 64 MB je naslovljiv po bajtih. Prikažite format naslovov glavnega pomnilnika [1].

Rešitev:
Skupno število linij v predpomnilniku $m = \frac{8\,\text{KB}}{16\,\text{B}} = \frac{2^{13}\text{B}}{2^4\text{B}} = 2^9 = 512$.

Skupno število setov v predpomnilniku $v = \frac{2^9}{2} = 2^8 = 256$, zato za označitev seta potrebujemo 8 bitov.

Glavni pomnilnik velikosti $64\,\text{MB} = 2^{26}\,\text{B}$ potrebuje 26-bitne naslove.

Število blokov v glavnem pomnilniku $= \frac{64\,\text{MB}}{16\,\text{B}} = \frac{2^{26}\text{B}}{2^4\text{B}} = 2^{22}$, zato za označitev številke bloka potrebujemo 22 bitov. Ti biti so sestavljeni iz bitov značke in bitov za označitev seta. Ker imamo 8 bitov za označitev seta, je polje z značko dolgo $22 - 8 = 14$ bitov.

Linije so dolge $16\,\text{B} = 2^4\,\text{B}$, zato potrebujemo 4 bite za označitev besede (bajta).

Naslov glavnega pomnilnika:

ZNAČKA	SET	BESEDA
14	8	4

4.3 Izračun parametrov predpomnilnika (3/8)

Za šestnajstiške pomnilniške naslove 111111_{16}, 666666_{16} in $BBBBBB_{16}$ prikažite naslednjo informacijo v šestnajstiški obliki [1]:

a) Vrednost v poljih za značko, linijo in besedo za predpomnilnik z neposredno preslikavo in format naslovov s spodnje slike.

ZNAČKA	LINIJA	BESEDA
8	14	2

b) Vrednost v poljih za značko in besedo za asociativni predpomnilnik in format naslovov s spodnje slike.

ZNAČKA	BESEDA
22	2

c) Vrednost v poljih za značko, set in besedo za dvakrat set-asociativni predpomnilnik in format naslovov s spodnje slike.

ZNAČKA	SET	BESEDA
9	13	2

Rešitev:
V spodnji tabeli so za vsako organizacijo predpomnilnika podani (šestnajstiški) pomnilniški naslovi, zapisani po bitih v dvojiškem številskem sistemu. Z modrim ozadjem so označeni biti značke, z zelenim biti linije, z rumenim biti besede in s sivim biti seta. Za vsako organizacijo predpomnilnika in podani pomnilniški naslov so pod dvojiškim zapisom naslova v šestnajstiškem številskem sistemu zapisane vrednosti v poljih za značko, linijo ali set in besedo.

	111111	666666	BBBBBB
a)	00010001000100010001001	01100110011001100110010	10111011101110111011011
	11 / 444 / 1	66 / 1999 / 2	BB / 2EEE / 3
b)	00010001000100010001001	01100110011001100110010	10111011101110111011011
	44444 / 1	199999 / 2	2EEEEE / 3
c)	00010001000100010001001	01100110011001100110010	10111011101110111011011
	22 / 444 / 1	CC / 1999 / 2	177 / EEE / 3

4.4 Izračun parametrov predpomnilnika (4/8)

Imamo 32-bitni mikroprocesor, ki ima v čipu 16 KB 4-krat set-asociativnega predpomnilnika. Linije predpomnilnika so dolge štiri 32-bitne besede. Prikažite format naslovov glavnega pomnilnika. Kam v predpomnilnik se preslika beseda s pomnilniške lokacije ABCDE8F8 [1]?

Rešitev:
Dolžina linij v predpomnilniku je $4 \cdot 32$ bitov $= 4 \cdot 4\,\mathrm{B} = 16\,\mathrm{B} = 2^4\,\mathrm{B}$.

Skupno število linij v predpomnilniku $m = \frac{2^{14}\mathrm{B}}{2^4\mathrm{B}} = 2^{10}$.

Skupno število setov v predpomnilniku $v = \frac{2^{10}\mathrm{B}}{2^2\mathrm{B}} = 2^8$.

ZNAČKA	SET	BESEDA
20	8	4

Pomnilniško lokacijo ABCDE8F8 zapišimo v dvojiškem številskem sistemu in z barvami označimo polja za značko, set in besedo.

10101011110011011110100011111000

S sivo barvo označeni biti določajo set, v katerega se preslika beseda iz pomnilnika. Biti 10001111_2 predstavljajo desetiško število 143. Beseda s pomnilniške lokacije ABCDE8F8 se torej preslika v katerokoli linijo seta 143.

4.5 Izračun parametrov predpomnilnika (5/8)

Procesor Intel 80486 ima v čip integriran skupni predpomnilnik za ukaze in podatke. Ima kapaciteto 8 KB, štirikrat set-asociativno organizacijo in linije dolžine štirih 32-bitnih besed. Predpomnilnik je organiziran v 128 setov. Za vsako linijo ima bit veljavnosti linije (V) in tri bite (B0, B1 in B2) za izvedbo algoritma LRU (Least Recently Used) za izbiro linije za zamenjavo. Ob zgrešitvi predpomnilnika procesor 80486 prebere 16 bajtov dolg blok iz glavnega pomnilnika v predpomnilnik. Pokažite, kako se interpretirajo različna polja pomnilniškega naslova [1].

Rešitev:
Linije so dolge $4 \cdot 32$ bitov $= 4 \cdot 4\,\mathrm{B} = 16\,\mathrm{B} = 2^4\,\mathrm{B}$, zato potrebujemo 4 bite za označitev besede (bajta).

Skupno število linij v predpomnilniku $m = \frac{8\,\mathrm{KB}}{16\,\mathrm{B}} = \frac{2^{13}\mathrm{B}}{2^4\mathrm{B}} = 2^9 = 512$.

Skupno število setov v predpomnilniku $v = \frac{2^9}{2^2} = 2^7 = 128$, zato za označitev seta potrebujemo 7 bitov.

Ker ima procesor 32-bitno naslovno vodilo, ima značka dolžino $32 - (7 + 4)$ bitov $=$ 21 bitov. To pomeni, da se v vsak set predpomnilnika preslika natanko 2^{21} blokov glavnega pomnilnika.

Naslov glavnega pomnilnika:

ZNAČKA	SET	BESEDA
21	7	4

Vsak set glavnega pomnilnika vsebuje tri bite LRU in štiri linije. Vsaka linija vsebuje štiri 32-bitne besede, bit veljavnosti V in 21-bitno značko.

4.6 Izračun parametrov predpomnilnika (6/8)

Zamislite si računalnik z bajtno naslovljivim glavnim pomnilnikom velikosti 2^{16} bajtov in velikostjo bloka 8 bajtov. Predpostavljajte, da računalnik uporablja predpomnilnik z 32 linijami z neposrednim preslikovanjem [1].

a) Kako je 16-bitni naslov pomnilnika razdeljen na značko, številko linije in številko bajta?

b) V katero linijo bi se shranili bajti iz vsakega izmed naslednjih naslovov?

0001 0001 0001 1011

1100 0011 0011 0100

1101 0000 0001 1101

1010 1010 1010 1010

c) Predpostavite, da je v predpomnilniku shranjen bajt z naslova 0001 1010 0001 1010. Kakšni so naslovi drugih bajtov, ki so shranjeni poleg njega?

d) Kolikšno je skupno število bajtov pomnilnika, ki so lahko shranjeni v predpomnilniku?

e) Zakaj je v predpomnilnik shranjena značka?

Rešitev:

a) Osem najbolj uteženih bitov označuje značko, pet srednjih bitov številko linije in trije najmanj uteženi biti številko bajta.

b)

0001000100011011	linija 3
1100001100110100	linija 6
1101000000011101	linija 3
1010101010101010	linija 21

c)

0001101000011010

Bajt z naslova 0001 1010 0001 1010 se preslika v linijo številka 3. V tej isti liniji je osem bajtov z naslovov od 0001 1010 0001 1000 do 0001 1010 0001 1111 (trije najmanj uteženi biti za izbiro bajta v liniji zavzamejo vse vrednosti od 000 do 111).

d) Predpomnilnik ima 32 linij po 8 bajtov, kar pomeni, da je lahko v njem skupno $32 \cdot 8 = 256$ bajtov glavnega pomnilnika.

e) V isto linijo predpomnilnika se preslikuje mnogo različnih blokov iz glavnega pomnilnika. Značka je potrebna, da označuje, kateri izmed teh blokov je trenutno v dani liniji predpomnilnika.

4.7 Izračun parametrov predpomnilnika (7/8)

Nek 2-krat set-asociativni predpomnilnik ima velikost linije oziroma bloka štirih 16-bitnih besed. V predpomnilnik lahko namestimo skupaj 4096 besed. Velikost glavnega pomnilnika je 64K×32 bitov. Prikažite, kako se tolmačijo procesorjevi naslovi [1].

Rešitev:
Velikost linije je $4 \cdot 16$ b $= 4$ besede $= 4 \cdot 2\,\mathrm{B} = 8\,\mathrm{B} = 2^3\,\mathrm{B}$. Če procesor dostopa do 16-bitne besede, potrebujemo dva bita za izbiro ene izmed štirih besed. Če procesor dostopa do bajta, potrebujemo poleg teh dveh bitov za izbiro besede še en bit za izbiro bajta znotraj izbrane besede. Za izbiro osebka v izbrani liniji predpomnilnika potrebujemo torej tri najmanj utežene bite procesorjevega naslova. Bita b_2b_1 izbereta besedo, pri dostopu do bajta pa bit b_0 izbere še bajt v tej besedi.

Velikost predpomnilnika je 4096 besed $= 2^{12}$ besed.

Število linij v predpomnilniku $m = \frac{2^{12}\text{besed}}{2^2\text{besed}} = 2^{10}$.

Število setov v predpomnilniku $v = \frac{2^{10}}{2} = 2^9$, zato potrebujemo 9 bitov za izbiro seta.

Velikost glavnega pomnilnika je 64K $\times$ 32 bitov $= 2^{16} \cdot 2^2\,\mathrm{B} = 2^{18}\,\mathrm{B}$. Naslovi pomnilnika so torej 18-bitni. Značka je potemtakem dolga $18 - (9 + 3) = 6$ bitov.

ZNAČKA	SET	BESEDA	
6	9	2	1

4.8 Izračun parametrov predpomnilnika (8/8)

Imate pomnilniški sistem, ki uporablja 32-bitni naslov za bajtno naslavljanje, in predpomnilnik, ki uporablja linije dolžine 64 bajtov [1].

a) Predpostavite predpomnilnik z neposrednim preslikovanjem, kjer je značka v naslovu dolga 20 bitov. Prikažite format naslova in določite naslednje parametre: število naslovljivih enot, število blokov v glavnem pomnilniku in število linij v predpomnilniku.

b) Predpostavite asociativni predpomnilnik. Prikažite format naslova in določite naslednje parametre: število naslovljivih enot, število blokov v glavnem pomnilniku in število linij v predpomnilniku.

c) Predpostavite 4-krat set-asociativni predpomnilnik, kjer je značka v naslovu dolga 9 bitov. Pokažite format naslova in določite naslednje parametre: število naslovljivih enot, število blokov v glavnem pomnilniku, število linij v setu, število setov v predpomnilniku in število linij v predpomnilniku.

Rešitev:

a) Dolžina linij v predpomnilniku je $64\,\mathrm{B} = 2^6\,\mathrm{B}$, torej potrebujemo 6 bitov za izbiro bajta (besede).

Število naslovljivih enot je 2^{32} B. Število bitov v naslovu za izbiro linije v predpomnilniku je $32-(20+6)=6$, torej je v predpomnilniku 2^6 linij. Število blokov v glavnem pomnilniku je $2^{(20+6)}=2^{26}$.

ZNAČKA	LINIJA	BESEDA
20	6	6

b) Dolžina linij v predpomnilniku je $64\,\mathrm{B} = 2^6\,\mathrm{B}$, torej potrebujemo 6 bitov za izbiro bajta (besede).

Število naslovljivih enot je $2^{32}\,\mathrm{B}$. Število bitov v naslovu za izbiro značke v predpomnilniku je $32-6=26$. Število blokov v glavnem pomnilniku je 2^{26}. Število linij v predpomnilniku ni določeno.

ZNAČKA	BESEDA
26	6

c) Dolžina linij v predpomnilniku je $64\,\mathrm{B} = 2^6\,\mathrm{B}$, torej potrebujemo 6 bitov za izbiro bajta (besede).

Število naslovljivih enot je $2^{32}\,\mathrm{B}$. Število bitov v naslovu za izbiro seta je $32-(9+6)=17$, torej je v predpomnilniku 2^{17} setov.

ZNAČKA	SET	BESEDA
9	17	6

4.9 Enačba za učinkovitost dostopa do predpomnilnika

Imamo dvonivojski pomnilnik s pomnilnikoma M_1 in M_2. M_1 je manjši, hitrejši in dražji (na bit) od pomnilnika M_2 (M_1 bi lahko bil na primer predpomnilnik, M_2 pa glavni pomnilnik) [1]. Če je

T_s povprečni (sistemski) čas dostopa,
T_1 čas dostopa do M_1,
T_2 čas dostopa do M_2 in
H verjetnost zadetka v M_1 (delež referenc, ki se najdejo v M_1),

lahko izračunamo povprečni čas dostopa kot

$$T_s = H \cdot T_1 + (1-H)\cdot(T_1+T_2).$$

a) Izrazite T_s s takim izrazom, da bo iz njega mogoče neposredno razbrati, za kolikšno vrednost je povprečni čas dostopa večji od T_1.

b) Izrazite količnik T_1/T_s, imenovan tudi kot *učinkovitost dostopa (access efficiency)*, ki je merilo o tem, kako blizu je povprečni čas dostopa (T_s) času dostopa do M_1 (T_1).

Rešitev:

a) $T_s = H \cdot T_1 + (1 - H) \cdot (T_1 + T_2) = H \cdot T_1 + T_1 - H \cdot T_1 + T_2 - H \cdot T_2$
$= T_1 + (1 - H) \cdot T_2$

b) $T_s = T_1 + (1 - H) \cdot T_2 / : T_1$

$$\frac{T_s}{T_1} = 1 + (1 - H) \cdot \frac{T_2}{T_1}$$
$$\frac{T_1}{T_s} = \frac{1}{(1 + (1 - H) \cdot \frac{T_2}{T_1})}$$

4.10 Pomnilniška hierarhija na N nivojih*

Zamislite si računalniški sistem s pomnilniško hierarhijo na N nivojih. Definirajmo naslednje oznake:

C_i = cena enega bita pomnilnika na pomnilniškem nivoju i,
S_i = velikost pomnilnika na nivoju i,
T_i = čas za dostop do besede na pomnilniškem nivoju i,
H_i = verjetnost zadetka v pomnilnik na nivoju i,
B_i = čas, potreben za prenos bloka podatkov iz pomnilniškega nivoja $(i + 1)$ v pomnilniški nivo i [1].

a) Izračunajte povprečno ceno C_s enega bita pomnilnika v tem sistemu.

b) Izračunajte povprečni (sistemski) čas dostopa T_s.

Rešitev:

a) Povprečno ceno enega bita pomnilnika v tem sistemu dobimo tako, da delimo vsoto produktov cene enega bita pomnilnika na nivoju i z velikostjo pomnilnika na nivoju i s skupno velikostjo vseh pomnilnikov v sistemu:

$$C_s = \frac{\sum_{i=1}^{N} C_i S_i}{\sum_{i=1}^{N} S_i}.$$

b) Pri izpeljavi povprečnega časa dostopa T_s uporabimo formulo za izračun pričakovane vrednosti oziroma matematičnega upanja E(X) diskretno porazdeljene slučajne spremenljivke X. E(X) je vsota produktov posamezne vrednosti slučajne spremenljivke in njene verjetnosti:

$$E(X) = \sum_{i=1}^{N} x_i \Pr[X = x_i].$$

V kontekstu N-nivojskega hierarhičnega pomnilnika je diskretna slučajna spremenljivka T_i, njena verjetnost pa H_i. Tako lahko zapišemo

$$T_s = \sum_{i=1}^{N} T_i H_i.$$

Ugotovimo lahko naslednje. Če je beseda v pomnilniku na nivoju 1 (v predpomnilniku), jo procesor prebere neposredno. Če je v pomnilniku na nivoju 2, ne pa na nivoju 1, se blok podatkov iz pomnilnika na nivoju 2 prenese v pomnilnik na nivoju 1 in od tam prebere. Za čas dostopa do pomnilnika na nivoju 2 imamo torej

$$T_2 = B_1 + T_1.$$

Izrazimo še T_3:

$$T_3 = B_2 + T_2 = B_1 + B_2 + T_1.$$

Če zgornji izraz posplošimo, dobimo

$$T_i = \sum_{j=1}^{i-1} B_j + T_1.$$

Zdaj ta izraz vstavimo v formulo za izračun matematičnega upanja za T_s:

$$\begin{aligned} T_s &= \sum_{i=1}^{N} \left(\sum_{j=1}^{i-1} B_j + T_1 \right) H_i = \sum_{i=1}^{N} \left(\sum_{j=1}^{i-1} B_j H_i + T_1 H_i \right) \\ &= \sum_{i=1}^{N} \sum_{j=1}^{i-1} (B_j H_i) + T_1 \sum_{i=1}^{N} H_i \\ &= \sum_{i=1}^{N} \sum_{j=1}^{i-1} (B_j H_i) + T_1, \end{aligned}$$

saj je

$$\sum_{i=1}^{N} H_i = 1.$$

4.11 Cena bita kombiniranega dvonivojskega pomnilnika

Imamo pomnilniški sistem z naslednjimi parametri: čas dostopa do predpomnilnika je 100 ns, cena predpomnilnika 10^{-4} USD/bit, čas dostopa do glavnega pomnilnika je 1.200 ns, cena glavnega pomnilnika 10^{-5} USD/bit.

a) Kolikšna je cena 1 MB glavnega pomnilnika?

b) Kolikšna je cena 1 MB glavnega pomnilnika z uporabo tehnologije predpomnilnika?

c) Kolikšna je verjetnost zadetka v predpomnilnik (H), če je povprečni čas dostopa do glavnega pomnilnika 10 % večji od časa dostopa do predpomnilnika?

d) Kolikšna je povprečna cena enega bita kombiniranega dvonivojskega pomnilnika, če ima predpomnilnik kapaciteto 4 KB, glavni pomnilnik pa kapaciteto 8 MB?

Rešitev:

a) $2^{20}\,\mathrm{B} \cdot 10^{-5}\,\mathrm{USD/bit} = 2^{20}\,\mathrm{B} \cdot 10^{-5} \cdot 8\,\mathrm{USD/B} = 83{,}89\,\mathrm{USD}$

b) $2^{20}\,\mathrm{B} \cdot 10^{-4}\,\mathrm{USD/bit} = 2^{20}\,\mathrm{B} \cdot 10^{-4} \cdot 8\,\mathrm{USD/B} = 838{,}86\,\mathrm{USD}$

c) $T_s = T_1 + (1 - H) \cdot T_2 = 1{,}1 \cdot T_1$

$0{,}1 \cdot T_1 = (1 - H) \cdot T_2$

Iz te enačbe izrazimo H:

$$H = \frac{T_2 - 0{,}1 \cdot T_1}{T_2} = \frac{1.200 - 0{,}1 \cdot 100}{1.200} = \frac{1.190}{1.200} = 0{,}9925.$$

d)

$$\begin{aligned} C_s &= \frac{C_1 S_1 + C_2 S_2}{S_1 + S_2} \\ &= \frac{10^{-4}\,\mathrm{USD/bit} \cdot 2^{12}\,\mathrm{B} + 10^{-5}\,\mathrm{USD/bit} \cdot 2^{23}\,\mathrm{B}}{2^{12}\,\mathrm{B} + 2^{23}\,\mathrm{B}} \\ &= \frac{10^{-4} \cdot 2^{12} + 10^{-5} \cdot 2^{23}}{2^{12} + 2^{23}}\,\mathrm{USD/bit} \\ &= 1{,}00439 \cdot 10^{-5}\,\mathrm{USD/bit} \end{aligned}$$

Kot vidimo, je povprečna cena enega bita kombiniranega dvonivojskega pomnilnika le malenkost višja od cene enega bita glavnega pomnilnika.

4.12 Preoblikovanje predpomnilnika

Zamislite si prvonivojski (L1) predpomnilnik s časom dostopa 1 ns in verjetnostjo zadetka H = 0,95. Predpostavite, da spremenimo zasnovo predpomnilnika (velikost in organizacijo) tako, da povečamo verjetnost zadetka na 0,97, hkrati pa povečamo čas dostopa na 1,5 ns. Kateri pogoj mora biti izpolnjen ob tej spremembi, da dobimo povečanje zmogljivosti predpomnilnika (skrajšanje povprečnega časa dostopa do glavnega pomnilnika) [1]?

Rešitev:

Ob začetnih pogojih je povprečni čas dostopa

$1{,}0\,\mathrm{ns} + 0{,}05 \cdot T_2$.

Ob spremenjenih pogojih je povprečni čas dostopa

$1{,}5\,\mathrm{ns} + 0{,}03 \cdot T_2$.

Za povečanje zmogljivosti mora veljati

$1{,}0\,\text{ns} + 0{,}05 \cdot T_2 > 1{,}5\,\text{ns} + 0{,}03 \cdot T_2.$

Rešitev neenačbe nam razkrije pogoj za povečanje zmogljivosti z novo organizacijo predpomnilnika: $\mathrm{T}_2 > 25\,\text{ns}$.

4.13 Računanje časa dostopa do predpomnilnika

Predpostavite enonivojski predpomnilnik s časom dostopa 2,5 ns, dolžino linije 64 bajtov in verjetnostjo zadetka H = 0,95. Glavni pomnilnik uporablja zmožnost prenosa blokov, ki ima čas dostopa do prve besede (4 bajtov) 50 ns in zatem 5 ns za vsako preostalo besedo [1].

a) Kakšen je čas dostopa ob zgrešitvi predpomnilnika? Predpostavite, da predpomnilnik počaka, dokler se linija ne včita iz glavnega pomnilnika, in potem ponovi dostop z zadetkom.

b) Predpostavite, da povečanje linije na 128 bajtov poveča H na 0,97. Ali to zmanjša povprečni čas dostopa do pomnilnika?

Rešitev:

a) Najprej poteče 2,5 ns, da se ugotovi dogodek zgrešitve predpomnilnika. Nato se zahtevana linija včita v predpomnilnik. Nazadnje je potrebnih še dodatnih 2,5 ns za včitanje zahtevane besede:

$T_{\text{miss}} = 2{,}5 + 50 + 15 \cdot 5 + 2{,}5 = 130\,\text{ns}.$

b) Vrednost T_{miss} iz točke **a)** je ekvivalentna vsoti časov T_1 in T_2, $T_1 + T_2$, pri čemer je T_1 čas dostopa do predpomnilnika in T_2 čas dostopa do glavnega pomnilnika. Povprečni čas dostopa do glavnega pomnilnika je tako

$T_s = H \cdot T_1 + (1 - H) \cdot (T_1 + T_2) = 0{,}95 \cdot 2,5 + 0{,}05 \cdot 130 = 8{,}875\,\text{ns}.$

Pri prenovljeni shemi predpomnilnika imamo:

$T_{\text{miss}} = 2{,}5 + 50 + 31 \cdot 5 + 2{,}5 = 210\,\text{ns}$

in

$T_s = H \cdot T_1 + (1 - H) \cdot (T_1 + T_2) = 0{,}97 \cdot 2{,}5 + 0{,}03 \cdot 210 = 8{,}725\,\text{ns}.$

Po prenovi organizacije predpomnilnika se povprečni čas dostopa do glavnega pomnilnika skrajša za $8{,}875\,\text{ns} - 8{,}725\,\text{ns} = 0{,}15\,\text{ns}$.

4.14 Povprečni čas dostopa do trinivojskega pomnilnika

Računalnik ima predpomnilnik, glavni pomnilnik in disk za izvedbo navideznega pomnilnika. Če je beseda, na katero se sklicuje procesor, v predpomnilniku, je potrebnih 20 ns, da jo doseže. Če je ni v predpomnilniku, je pa v glavnem pomnilniku,

je potrebnih 60 ns za njeno naložitev v predpomnilnik, nato pa se ponovi sklic do nje. Če besede ni v glavnem pomnilniku, je potrebnih 12 ms za njen vnos z diska v glavni pomnilnik, nato 60 ns za njeno kopiranje v predpomnilnik, nazadnje pa še ponovitev sklica nanjo v predpomnilnik. Verjetnost zadetka v predpomnilnik je 0,9, verjetnost zadetka v glavni pomnilnik pa 0,6. Kolikšen je povprečni čas za dostop do sklicevane besede v tem sistemu [1]?

Rešitev:

Definirajmo najprej uporabljene oznake in njihove vrednosti:
$H_1 = 0{,}9$: verjetnost zadetka v predpomnilnik,
$H_2 = 0{,}6$: verjetnost zadetka v glavni pomnilnik,
$T_1 = 20\,\text{ns}$: čas za dostop do besede v predpomnilniku,
$T_2 = 60\,\text{ns}$: čas za prenos besede (v bloku) iz glavnega pomnilnika v predpomnilnik,
$T_3 = 12\,\text{ms}$: čas za včitanje besede (znotraj sektorja) iz diska v glavni pomnilnik.

Obravnavati moramo tri primere:
1. sklic je na besedo v predpomnilniku,
2. sklic je na besedo, ki je ni v predpomnilniku, je pa v glavnem pomnilniku in
3. sklic je na besedo, ki je ni niti v predpomnilniku niti v glavnem pomnilniku, pač pa je na disku.

Povprečni čas T_a za dostop do sklicevane besede v tem sistemu je:

$$\begin{aligned} T_a &= H_1 \cdot T_1 + (1 - H_1) \cdot H_2 \cdot (T_2 + T_1) + (1 - H_1) \cdot (1 - H_2) \cdot (T_3 + T_2 + T_1) \\ &= 0{,}9 \cdot 20 + (1 - 0{,}9) \cdot 0{,}6 \cdot (60 + 20) + (1 - 0,9) \cdot (1 - 0{,}6) \cdot (12 \cdot 10^6 + 60 + 20) \\ &= 0{,}9 \cdot 20 + 0{,}06 \cdot 80 + 0{,}04 \cdot 12.000.080 \\ &= 480.026\,\text{ns} = 480{,}026\,\mu\text{s}. \end{aligned}$$

4.15 Predpomnilnik pri mikroprocesorju Motorola 68020

Pri mikroprocesorju Motorola 68020 traja dostop do predpomnilnika dva urina cikla. Dostop do podatka v glavnem pomnilniku prek vodila k procesorju traja tri urine cikle, če ni vrivanja čakalnih stanj. Podatki se dostavijo procesorju hkrati z dostavo v predpomnilnik [1].

a) Izračunajte povprečno dolžino pomnilniškega cikla, če je verjetnost zadetka v predpomnilnik 0,9 in procesor dela na frekvenci 16,67 MHz.

b) Ponovite izračun, če mora procesor v vsak pomnilniški cikel vriniti dve periodi ure. Kakšna ugotovitev lahko sledi iz rezultatov?

Rešitev:

a) Pri frekvenci 16,67 MHz traja perioda ure 60 ns. Dostop do predpomnilnika traja dve periodi ure, kar znaša 120 ns, dostop do glavnega pomnilnika pa traja tri periode ure, kar je 180 ns. Povprečna dolžina pomnilniškega cikla znaša torej $0{,}9 \cdot 120 + 0{,}1 \cdot 180 = 126$ ns.

b) Vsak dostop do glavnega pomnilnika je podaljšan za dve periodi ure, torej za 120 ns. Povprečna dolžina pomnilniškega cikla zdaj znaša $0{,}9 \cdot 120 + 0{,}1 \cdot 300 = 138$ ns. Čeprav se čas dostopa do pomnilnika poveča za 120 ns, se povprečni čas dostopa poveča samo za 12 ns, saj je v povprečju samo vsak deseti dostop procesorja dostop v glavni pomnilnik.

4.16 Vpliv ukaznega predpomnilnika na izkoriščenost vodila

Zamislite si procesor s časom pomnilniškega cikla 300 ns in hitrostjo izvajanja ukazov 1 MIPS. Vsak ukaz v povprečju zahteva en pomnilniški cikel na vodilu za včitanje ukaza in enega za dostop do operanda, ki ga ukaz vpeljuje [1].

a) Izračunajte izkoriščenost vodila.

b) Predpostavite, da je procesor opremljen z ukaznim predpomnilnikom, ki ima verjetnost zadetka 0,5. Določite vpliv na izkoriščenost vodila.

Rešitev:

a) Za procesor s hitrostjo izvajanja ukazov 1 MIPS povprečni ukaz porabi 1.000 ns za včitanje in izvedbo. V povprečju traja ukaz dva cikla na vodilu, kar znaša 600 ns, tako da je izkoriščenost vodila $\frac{600}{1.000} = 0{,}6$.

b) Samo za polovico ukazov se ukaz včita prek vodila. Izkoriščenost vodila je zdaj $\frac{150+300}{1.000} = 0{,}45$. To zmanjša čas čakanja za druge potencialne gospodarje na vodilu, kot so krmilniki za neposredni dostop do pomnilnika (DMA) in drugi mikroprocesorji.

4.17 Preslikave blokov v predpomnilnika različnih tipov

Imamo glavni pomnilnik, sestavljen iz 32 blokov, ki so označeni s številkami od 0 do 31, in predpomnilnik z osmimi linijami, označenimi od 0 do 7 [1].

a) Kateri bloki glavnega pomnilnika tekmujejo za preslikavo v linijo 2 predpomnilnika, če predpomnilnik uporablja neposredno preslikovanje?

b) V katere linije predpomnilnika se lahko preslika blok 31 glavnega pomnilnika, če predpomnilnik uporablja 4-krat set-asociativno preslikovanje?

c) CPE se sklicuje na bloke glavnega pomnilnika po takšnem zaporedju, kot je prikazano na spodnji sliki. V začetku je predpomnilnik prazen. Pri katerem sklicu se mora vsebina predpomnilnika nujno razlikovati za neposredno preslikovanje in 4-krat set-asociativno preslikovanje?

vrstni red sklicev	1	2	3	4	5	6	7	8
sklic na blok	0	15	18	5	1	13	15	26

Rešitev:

a) Če ima predpomnilnik m linij, se blok i glavnega pomnilnika preslika v linijo j predpomnilnika po naslednji enačbi: $j = i \mod m$. Če za j vstavimo vrednost 2 in za m vrednost 8, dobimo enačbo $2 = i \mod 8$. Rešitev enačbe so številke tistih blokov glavnega pomnilnika, ki imajo pri deljenju z vrednostjo 8 ostanek 2. Izmed blokov s številkami od 0 do 31 so to bloki 2, 10, 18 in 26.

b) Število setov v predpomnilniku izračunamo tako, da število linij delimo s $k = 4$: $v = \frac{2^3}{2^2} = 2$. Predpomnilnik je torej sestavljen iz dveh setov, označenih s številkama 0 in 1. Če ima predpomnilnik v setov, se blok i glavnega pomnilnika preslika v set j predpomnilnika po naslednji enačbi: $j = i \mod v$. Če za j vstavimo vrednost 31 in za v vrednost 2, dobimo enačbo $j = 31 \mod 2$. Bloki s sodimi številkami se preslikujejo v set 0, oni z lihimi številkami pa v set 1. Blok 31 je lih, zato se lahko preslika v katerokoli linijo seta 1, torej v linije 4, 5, 6 ali 7.

c) Pri neposrednem preslikovanju se blok i glavnega pomnilnika preslika v linijo $j = i \mod 8$, torej v linijo s tisto številko, ki vrne pri deljenju številke bloka glavnega pomnilnika s številom 8 ostanek 2. V spodnji tabeli so v tretji vrstici prikazane številke linij v predpomnilniku, v katere se preslika posamezni blok glavnega pomnilnika, na katerega se sklicuje CPE.

vrstni red sklicev	1	2	3	4	5	6	7	8
sklic na blok	0	15	18	5	1	13	15	26
številka linije	0	7	2	5	1	5	7	2

Pri 4-krat set-asociativnem predpomnilniku je več prostosti, kam v izbrani set se lahko namesti blok, na katerega se sklicuje CPE. Zdaj bomo oponašali izbiro linije v predpomnilniku kot v primeru neposrednega preslikovanja, dokler bo to mogoče.

vrstni red sklicev	1	2	3	4	5	6	7	8
sklic na blok	0	15	18	5	1	13	15	26
številka linije	0	7	2	5	x			

Set 0 je sestavljen iz linij 0, 1, 2 in 3, set 1 pa iz linij 4, 5, 6 in 7. Blok 0 lahko namestimo v linijo 0 seta 0, blok 15 v linijo 7 seta 1, blok 18 v linijo 2 seta 0, blok 5 pa v linijo 5 seta 1. Vse do tod lahko preslikamo bloke glavnega pomnilnika v iste linije kot pri neposrednem preslikovanju. Od tod naprej pa ne gre več, saj moramo blok 1 preslikati v set 1 (tam sta prosti še liniji 4 in 6), pri neposrednem preslikovanju pa se je blok 1 preslikal v linijo 1. Tako lahko zaključimo, da se mora vsebina predpomnilnika za neposredno in 4-krat set-asociativno preslikavo nujno razlikovati pri petem sklicu.

5 Notranji pomnilnik

5.1 Izvedba 256 KB pomnilnika za 8-bitni mikroprocesor

Za 8-bitni mikroprocesor morate izvesti 256 KB pomnilnika [3].

a) Koliko dinamičnih pomnilnikov z naključnim dostopom (DRAM-ov) 64K×1 potrebujete za implementacijo 256 KB pomnilnika za 8-bitni mikroprocesor?

b) Koliko čipov pa potrebujete, če uporabite DRAM-e 256K×1.

c) Koliko znaša najmanjši inkrement v pomnilniški kapaciteti za obe izvedbi? Primerjajte tudi porabo moči, površino zasedenega prostora na tiskanini, zanesljivost in ceno komponent.

Rešitev:

a) Za implementacijo 256 KB pomnilnika z DRAM-i 64K×1 potrebujemo štiri banke po osem čipov v vsaki, torej skupno 32 DRAM-ov.

b) Enako kapaciteto lahko dosežemo samo z osmimi čipi 256K×1.

c) Velikost najmanjšega inkrementa pomnilniške kapacitete ustreza tisti, dobljeni z 8 DRAM-i. Za čipe 64K×1 je to 64 KB, za čipe 256K×1 pa 256 KB. Problem pri DRAM-ih 256K×1 je torej granularnost. Če potrebujemo na primer le 64 KB pomnilnika, je implementacija z DRAM-i 256K×1 neprimerna. Pomnilniške kapacitete ni mogoče nadgrajevati v inkrementih, manjših od 256 KB. Z DRAM-i večje kapacitete prihranimo pri porabi moči in prostoru na tiskanini. Pridobimo tudi na zanesljivosti zaradi manjšega števila čipov in znižanja stroškov za čipe, saj je cena DRAM-a 256K×1 precej manjša od cene štirih DRAM-ov 64K×1.

5.2 Izvedba 512 KB pomnilnika za 8-bitni mikroprocesor

Načrtovalec namerava za izvedbo 512 KB pomnilnika za 8-bitni mikroprocesor uporabiti 256 Kb (kilobitne) DRAM-e [3].

a) Koliko DRAM-ov 256K×1 je potrebnih?

b) Koliko DRAM-ov pa je potrebnih, če so DRAM-i organizirani ×4?

c) Primerjajte obe implementaciji v smislu porabe moči in prostora na tiskanini.

Rešitev:

a) Osem DRAM-ov 256K×1 zagotovi 256 KB pomnilnika, zato potrebujemo dve banki po osem DRAM-ov – torej skupno 16 čipov.

b) Po drugi strani dva DRAM-a 64K×4 zagotavljata 64 KB pomnilnika. Potrebujemo torej osem bank po dva DRAM-a v vsaki. Spet potrebujemo skupaj 16 čipov.

c) Zaključimo lahko, da za tako velik pomnilnik ostane število čipov enako, ne glede na organizacijo DRAM-ov. Imamo pa z DRAM-i 64K×4 bolj drobno granulacijo. Poraba moči je tudi nižja, ker je vsaka banka sestavljena iz samo dveh čipov. Slaba stran pa je seveda poraba prostora. DRAM-i 64K×4 so izvedeni v 18-nožičnih ohišjih in zato porabijo več prostora na tiskanini.

5.3 Osveževanje DRAM pomnilnika (1/2)

Imamo DRAM, ki potrebuje osveževalni cikel 64-krat na milisekundo. Vsaka operacija osveževanja traja 150 ns, medtem ko traja pomnilniški cikel 250 ns. Kolikšen odstotek skupnega časa delovanja pomnilnika je treba nameniti osveževanju [1]?

Rešitev:
V 1 ms je čas, namenjen osveževanju, $64 \cdot 150\,\text{ns} = 9.600\,\text{ns}$. Delež časa, namenjenega osveževanju pomnilnika, je $\frac{9{,}6 \cdot 10^{\check{}6}\,\text{s}}{10^{\check{}3}\,\text{s}} = 0{,}0096$ oziroma 0,96 %.

5.4 Osveževanje DRAM pomnilnika (2/2)

Pomnilnik nekega mikroračunalnika je zgrajen iz DRAM-ov 64K×1. Po podatkih iz priročnika je polje celic v DRAM-u organizirano v 256 vrsticah. Vsaka vrstica mora biti osveževana s periodo, ki ni daljša od 4 ms. Predpostavite, da osvežujemo pomnilnik strogo periodično. Kakšna je časovna perioda med zaporednima osveževalnima zahtevama? Kako dolg števnik z osveževalnimi naslovi potrebujemo [3]?

Rešitev:
Perioda osveževanja (od vrstice do vrstice) je lahko maksimalno $\frac{4.000}{256} = 15{,}625\,\mu\text{s}$. Ker imamo skupno 256 vrstic (to so vrstični naslovi 0–255), potrebujemo 8-bitni števnik ($256 = 2^8$).

5.5 Izračun stopnje odpovedi 512 KB pomnilnika

512 KB pomnilnika 16-bitnega mikroprocesorja je izvedenega z DRAM-i 64K×1 na eni tiskani plošči. Razen pomnilniških čipov vsebuje še 11 čipov srednje stopnje integracije (MSI) in 25 čipov nizke stopnje integracije (SSI). Izračunajte stopnjo odpovedi sistema, če je na tiskanini 10 uporov, 100 keramičnih kondenzatorjev in so na njej še štirje tantalovi kondenzatorji. Predpostavite tipične stopnje odpovedi v spodnji tabeli. FIT (Failures in Time) pomeni število odpovedi v času 10^9 ur [3].

Tip komponente	Tipična stopnja odpovedi (FIT)
upor	1
dioda	1
čip SSI	10
keramični kondenzator	10
tantalov kondenzator	20
čip MSI	50
tiskanina	500
DRAM 64K×1	1.200

Rešitev:
Vsaka pomnilniška banka zahteva 16 DRAM-ov in zagotavlja 128 KB. Za skupno 512 KB potrebujemo štiri banke, to je 64 DRAM-ov. Stopnja odpovedi pomnilniškega polja je $1.200 \cdot 64 = 76.800$ FIT. Preostale komponente prispevajo $11 \cdot 50 + 25 \cdot 10 + 10 \cdot 1 + 100 \cdot 10 + 4 \cdot 20 + 1 \cdot 500 = 2.390$ FIT. Iz tega sledi, da je stopnja odpovedi celotne pomnilniške plošče $76.800 + 2.390 = 79.190$ FIT. Vidimo, da pomnilniški čipi prispevajo okrog 97 % odpovedi.

5.6 Izračun MTBF-ja

Večanje števila komponent skrajšuje MTBF v sistemu [3].

a) Izračunajte MTBF (povprečni čas med odpovedima) za DRAM 64K×1 iz naloge **5.5**.

b) Ponovite izračun za pomnilniško ploščo kapacitete 512 KB iste naloge.

c) Predpostavite velik sistem z 32 mikroračunalniki. Vsak mikroračunalnik je opremljen z eno tako ploščo. Kakšen je MTBF v tem sistemu?

Rešitev:

a) MTBF DRAM-a je $\frac{10^9}{1.200} = 830.000$ ur, to je 95 let.

b) Za pomnilniško ploščo imamo MTBF $\frac{10^9}{79.190} = 12.600$ ur, to je 17 mesecev. V povprečju lahko pričakujemo odpoved pomnilniške plošče vsako poldrugo leto.

c) Pomnilnik celotnega sistema znaša $512 \cdot 32 = 16$ MB (32 pomnilniških plošč). Njegova stopnja odpovedi bo 32-krat večja od tiste za eno pomnilniško ploščo, ali kar je ekvivalentno, njegov MTBF bo 32-krat krajši. Torej, $\frac{12.600}{32} = 384$ ur, kar znese samo 16,4 dneva. Torej lahko pričakujemo odpoved pomnilnika nekje v sistemu približno vsak drugi teden.

5.7 MTBF brez odkrivanja enojnih napak in z njim

Sistem ima 8-bitni mikroprocesor s 128 KB pomnilnika, ki je izveden z DRAM-i 64K x 1 s stopnjo odpovedi 1.200 FIT. Narava aplikacije je taka, da se odpovedi tolerirajo, če se le pravočasno odkrijejo in se sistem ustavi. Odločeno je bilo, da dodamo en paritetni bit na bajt za odkrivanje napak [3].

a) Kako pogosto bi pomnilnik odpovedoval, preden dodamo pariteto? Predpostavite, da so vse odpovedi zaradi enojnih napak.

b) Kako pogosto se bo sistem ustavljal zaradi odpovedi pomnilnika? Ignorirajte odpovedi zaradi drugih komponent, kot so pomnilniška vmesniška vezja.

Rešitev:

a) V nalogi **5.6** smo izračunali, da znaša MTBF za en DRAM 64K x 1 95 let. Za skupno 128 KB potrebujemo 16 DRAM-ov organiziranih v dve banki. Zaradi tega se MTBF pomnilnika zmanjša na $\frac{95}{16}$, to je okrog 6 let.

b) Za odgovor na to vprašanje moramo upoštevati število dodanih DRAM-ov za izvedbo paritete, kar zahteva en DRAM 64K × 1 na banko. Skupno število DRAM-ov je torej 18, zato se MTBF pomnilnika zmanjša na $\frac{95}{18}$, to je okrog 5 let. Prednost je seveda ta, da je napake v pomnilniku zdaj mogoče odkriti. Ker so vse odpovedi po predpostavki zaradi enojnih pomnilniških napak, se vse odkrijejo. Iz tega sledi, da bodo napake v pomnilniku v povprečju ustavile sistem vsakih pet let.

5.8 MTBF brez popravljanja enojnih napak in z njim

Pomnilnik s skupno kapaciteto 1 MB za 16-bitni mikroprocesor je izveden z DRAM-i 256K×1 [3].

a) Izračunajte njegov MTBF, če predpostavljate 2.000 FIT za vsak DRAM.

b) Imejmo napravo za odkrivanje in popravljanje napak (EDAC), ki zmore popravljati enojne napake. Koliko s tem povečamo zanesljivost, če se 98 % napak pojavlja na posamičnih bitih?

Rešitev:

a) Vsaka banka zahteva 16 DRAM-ov in zagotavlja 512 KB pomnilnika. Skupno število zahtevanih čipov je torej 32. Skupna stopnja odpovedi (če razen DRAM-ov zanemarimo vse preostale komponente) je $2.000 \cdot 32 = 64.000$ FIT. To ustreza MTBF-ju $\frac{10^9}{64.000} = 15.625$ ur $= 22$ mesecev.

b) Vsaka pomnilniška banka zahteva šest dodatnih DRAM-ov za hranjenje testnih bitov. Skupno število čipov postane zdaj $22 \cdot 2 = 44$. Stopnja odpovedi znaša $2.000 \cdot 44 = 88.000$ FIT. Vendar zdaj samo večkratne napake prispevajo k odpovedim (enojne napake se popravijo), zato se efektivna stopnja odpovedi pomnilnika zmanjša na $88.000 \cdot 0{,}02 = 1.760$ FIT. To ustreza MTBF-ju $\frac{10^9}{1.760} = 568.180$ ur $= 790$ mesecev. Zanesljivost pomnilnika se je torej povečala za faktor $\frac{790}{22} = 36$. Iz tega rezultata je razvidno, da se zanesljivost dramatično poveča že pri popravljanju enojnih napak.

5.9 Poraba moči pri odkrivanju/popravljanju napak

EDAC vpliva na porabo moči in ponavadi tudi na hitrost sistema. Predpostavite 16-bitni mikroprocesor z 1 MB pomnilnika [3].

a) Za koliko odstotkov se poveča poraba moči, če dodamo zmožnost odkrivanja dvojnih/popravljanja enojnih napak?

b) Naj bo stanje CPE dolgo 100 ns in naj pomnilniški čas dostopa zagotovi, da bo podatek na voljo 20 ns pred včitanjem. Koliko čakalnih stanj bi morala CPE vriniti v vsak pomnilniški cikel, če dana naprava EDAC vnaša zakasnitev 40 ns?

Rešitev:

a) Poraba moči se poveča zaradi dodatne kapacitete pomnilnika, ki jo potrebujemo. Ker imamo 16-bitne besede, moramo dodati 6 testnih bitov in poraba moči se poveča za $\frac{22-16}{16} \cdot 100\ \% = 37{,}5\ \%$.

b) Podatki so veljavni 20 ns pred časom včitanja, vendar ta čas ne more kompenzirati dodatne zakasnitve 40 ns, če dodamo napravo EDAC. CPE mora vriniti eno čakalno stanje na pomnilniški cikel. To podaljša pomnilniški cikel za 100 ns.

5.10 Hammingov kod (7,4)

V pomnilniku želimo popravljati enobitne napake, zato moramo k m podatkovnim bitom dodati r testnih (paritetnih) bitov in nato celotno kodno besedo dolžine $n = m + r$ bitov vpisati v pomnilnik.

a) Koliko testnih bitov morate dodati, da boste po prebrani kodni besedi iz pomnilnika lahko izračunali sindrom, ki bo pokazal na bitno mesto v kodni besedi, na katerem se je pojavila napaka?

b) Izračunajte število potrebnih testnih bitov za $m = 4$ in napišite, kako ta Hammingov kod označujemo.

Rešitev:

a) Sindrom je dolg r bitov, torej ima enako dolžino kot znaša število testnih bitov. Število vseh možnih vrednosti sindroma je torej 2^r. Če imajo vsi biti sindroma vrednost 0, pomeni, da v pomnilniku v kodni besedi ni prišlo do nobene napake. Potemtakem ostane še $2^r - 1$ vrednosti sindroma, ki morajo pokazati na enega izmed bitov v $m+r$ bitov dolgi kodni besedi, saj lahko pride do enojne napake bodisi na podatkovnem ali testnem bitu. Število neničelnih vrednosti sindroma mora biti torej vsaj tolikšno, kolikor znaša dolžina kodne besede:

$$2^r - 1 \geq m + r \Rightarrow m + r + 1 \leq 2^r$$

b) V izpeljano neenačbo iz točke **a)** za m vstavimo vrednost 4 in dobimo

$$4 + r + 1 \leq 2^r \Rightarrow 5 + r \leq 2^r \Rightarrow r = 3.$$

Za popravljanje enobitne napake pri štirih podatkovnih bitih potrebujemo tri testne bite. Hammingov kod, ki vsebuje štiri podatkovne in tri testne bite, skupaj pa je kodna beseda dolga 7 bitov, označujem kot dvojico $(n, m) = (7, 4)$, kjer prvi element dvojice predstavlja skupno število bitov kodne besede v pomnilniku, drugi pa število podatkovnih bitov.

5.11 Matriki G in H Hammingovega koda (7,4)

Za popravljanje enojnih napak v pomnilniku uporabljamo Hammingov kod (n, m).

a) V tabeli prikažite, na katerih bitnih mestih se morajo nahajati testni (paritetni) in podatkovni biti, da bo sindrom pokazal na bitno mesto enojne napake. V tabeli tudi označite, nad katerimi podatkovnimi biti se računa posamezni testni bit.

b) Za prve štiri testne bite (p_1, p_2, p_3, p_4) napišite enačbe, iz katerih bo razvidno, nad katerimi podatkovnimi biti se posamezni testni bit računa.

c) S pomočjo tabele iz točke **a)** razberite vsebino matrike **G** za generiranje kodne besede $(7, 4)$, ki se zapiše v pomnilnik, in matriko zapišite.

d) S pomočjo tabele iz točke **a)** razberite vsebino matrike **H** za preverjanje paritete kodne besede $(7, 4)$, ki se prebere iz pomnilnika, in matriko zapišite.

e) S pomočjo tabele iz točke **a)** razberite vsebino matrike **R** za dekodiranje podatkovnih bitov iz kodne besede v pomnilniku po ugotovitvi, da je kodna beseda brez napak ali pa je popravljena po odkritju enojne napake.

Rešitev:

a)

Bitno mesto	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Kodna beseda	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8
p_1	×		×		×		×		×		×	
p_2		×	×			×	×			×	×	
p_3				×	×	×	×					×
p_4								×	×	×	×	×

$\ldots$ $\quad$ $\vdots$

Testni biti p_1, p_2, p_3, $p_4 \ldots$ se morajo nahajati na bitnih mestih potenc števila 2, to je na bitnih mestih 1, 2, 4, 8... Podatkovni biti d_1, d_2, d_3, $d_4 \ldots$ zasedajo po vrstnem redu vsa bitna mesta, ki niso potence števila 2. Tabela je narisana samo do podatkovne besede dolžine osem bitov, ki potrebuje štiri testne bite. Kot je prikazano s tremi horizontalnimi in tremi vertikalnimi pikami, se lahko tabela po potrebi razširi v obe dimenziji.

Podatkovne bite, nad katerimi se izračuna posamezni testni bit, najlažje določimo na naslednji način. V stolpcu za vsako bitno mesto s simboli × označimo binarno kodo tega bitnega mesta. Simbol × v tem kodu predstavlja logično vrednost 1, prazno mesto brez simbola pa logično vrednost 0. Najbolj uteženi bit v tej kodi je v spodnji celici stolpca, najmanj uteženi bit pa v zgornji. V narisanem delu tabele s štirimi paritetnimi biti dobimo kode bitnih mest 0001, 0010, 0011... Iz te tabele zdaj z lahkoto odčitamo podatkovne bite, nad katerimi se računa vrednost posameznega testnega bita. Pariteta za dani testni bit v vrstici se računa nad tistimi podatkovnimi biti, ki imajo v tej vrstici simbol ×.

b) Iz tabele, sestavljene v točki **a)**, lahko zapišemo izraze za izračun prvih štirih testnih bitov, kjer simbol $\oplus$ označuje operacijo XOR (ekskluzivni ALI).

$p_1 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7$
$p_2 = d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7$
$p_3 = d_2 \oplus d_3 \oplus d_4 \oplus d_8$
$p_4 = d_5 \oplus d_6 \oplus d_7 \oplus d_8$

Če ima v enačbi za izračun testnega bita sodo število podatkovnih bitov vrednost 1, bo testni bit imel vrednost 0, sicer bo imel vrednost 1. Alternativa za uporabo operacije $\oplus$ je navadno seštevanje vrednosti posameznih podatkovnih bitov v enačbi, nato deljenje vsote z 2 in ugotavljanje, kakšen je ostanek. Če je ostanek 0, je imelo sodo število podatkovnih bitov vrednost 1 in testni bit bo imel vrednost 0, če pa je ostanek 1, je imelo liho število podatkovnih bitov vrednost 1, testni bit pa bo imel vrednost 1.

c) **G**: Generatorska matrika linearnega koda (7,4):

$$\mathbf{G} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Generatorska matrika **G** linearnega koda (7,4) vsebuje sedem vrstic in štiri stolpce. Vrstice matrike ustrezajo bitnim mestom kode od 1 do 7, stolpci pa podatkovnim bitom d_1, d_2, d_3 in d_4. V vrsticah 1, 2 in 4, ki ustrezajo bitnim mestom, kjer so v kodi testni biti p_1, p_2 in p_3, vrednost 1 pomeni, da je podatkovni bit vključen v izračun paritete, vrednost 0 pa, da ni vključen. Vrstice matrike, ki niso potence števila 2, ustrezajo enemu podatkovnemu bitu. Ta je v vrstici označen z vrednostjo 1, na drugih treh mestih v vrstici pa so vrednosti 0.

d) **H**: Matrika za preverjanje paritete kodne besede (7,4):

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Matriko **H** za preverjanje paritete kodne besede $(7, 4)$ dobimo iz zgornjega levega dela tabele iz točke **a)**, tako da pogledamo pravokotnik, ki ga tvorijo vrstice od p_1 do p_3 in stolpci od bitnega mesta 1 do bitnega mesta 7. Matrika **H** ima tako tri vrstice in sedem stolpcev. Na mestu, kjer je v navedenem pravokotniku simbol $\times$, v matriko vpišemo vrednost 1, na drugih mestih pa vrednost 0.

e) **R**: Matrika za dekodiranje podatkovnih bitov iz kodne besede brez napak ali popravljene enojne napake za kod (7,4):

$$\mathbf{R} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Matrika **R** ima štiri vrstice in sedem stolpcev. V i-ti vrstici je enica na tistem bitnem mestu, na katerem se nahaja podatkovni bit d_i, na vseh drugih mestih so vrednosti 0.

5.12 Izračun sindroma za Hammingov kod (7,4)

Predpostavite, da za popravljanje enobitnih napak v pomnilniku uporabljate Hammingov linearni kod (7,4).

a) Za splošni 4-bitni podatkovni vektor $\boldsymbol{p} = [d_1 d_2 d_3 d_4]^{\mathrm{T}}$ (ta oznaka pomeni, da je vrstični vektor transponiran in dejansko predstavlja stolpični vektor) prikažite, kako se izračuna 7-bitna kodna beseda s tremi testnimi (paritetnimi) in štirimi podatkovnimi biti.

b) Izračunajte 7-bitno kodno besedo za podatkovni vektor $\boldsymbol{p} = [1011]^{\mathrm{T}}$.

c) Izračunajte sindrom, če v pomnilniku ni prišlo do nobene napake.

d) Za primer iz točke **c)** izvedite dekodiranje podatkovnih bitov iz kodne besede v pomnilniku.

e) Izračunajte sindrom, če je prišlo v pomnilniku do napake na podatkovnem bitu d_2.

f) Izračunajte sindrom, če je prišlo v pomnilniku do napake na podatkovnem bitu d_1.

g) Izračunajte sindrom, če je prišlo v pomnilniku do napake na testnem bitu p_2.

h) Izračunajte sindrom, če je prišlo v pomnilniku do napake na podatkovnem bitu d_1 in napake na podatkovnem bitu d_2. Ali lahko iz izračunanega sindroma odkrijete, da je prišlo do dvojne napake, torej do napake na dveh različnih bitih?

Pri izračunih uporabite generatorsko matriko **G**, matriko **H** za preverjanje paritete kodne besede in matriko **R** za dekodiranje podatkovnih bitov za kod (7,4), ki so bile predstavljene v problemu **5.11**.

Rešitev:

a) Zapišimo najprej splošno 4-bitno podatkovno besedo v obliki stolpičnega vektorja $\boldsymbol{p}$:

$$\boldsymbol{p} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

Zdaj s pomočjo generatorske matrike **G** izračunamo splošno 7-bitno kodno besedo $\boldsymbol{x}$:

$$\boldsymbol{x} = \mathbf{G} \times \boldsymbol{p} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix} = \begin{bmatrix} d_1 \oplus d_2 \oplus 0 \oplus d_4 \\ d_1 \oplus 0 \oplus d_3 \oplus d_4 \\ d_1 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus d_2 \oplus d_3 \oplus d_4 \\ 0 \oplus d_2 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus d_3 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus d_4 \end{bmatrix} = \begin{bmatrix} d_1 \oplus d_2 \oplus d_4 \\ d_1 \oplus d_3 \oplus d_4 \\ d_1 \\ d_2 \oplus d_3 \oplus d_4 \\ d_2 \\ d_3 \\ d_4 \end{bmatrix}$$

b) Za konkretni 4-bitni podatkovni vektor $\boldsymbol{p} = [d_1 d_2 d_3 d_4]^{\mathrm{T}} = [1011]^{\mathrm{T}}$ zdaj izračunajmo 7-bitno kodno besedo $\boldsymbol{x} = [p_1 p_2 d_1 p_3 d_2 d_3 d_4]^{\mathrm{T}}$, ki se vpiše v pomnilnik:

$$\boldsymbol{x} = \mathbf{G} \cdot \boldsymbol{p} = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 0 \oplus 0 \oplus 1 \\ 1 \oplus 0 \oplus 1 \oplus 1 \\ 1 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 1 \oplus 1 \\ 0 \oplus 0 \oplus 0 \oplus 0 \\ 0 \oplus 0 \oplus 1 \oplus 0 \\ 0 \oplus 0 \oplus 0 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

c) Prebrano kodno besedo iz pomnilnika označimo z $\boldsymbol{r}$. Če v pomnilniku ni bilo napak, je prebrana kodna beseda enaka vpisani: $\boldsymbol{p} = \boldsymbol{x}$.

Izračun sindroma $\boldsymbol{z}$:

$$\boldsymbol{z} = \mathbf{H} \cdot \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Ker v pomnilniku ni prišlo do nobene napake, je izračunani sindrom enak 0.

d) Ko ugotovimo, da je kodna beseda v pomnilniku brez napak, lahko podatkovne bite iz nje dekodiramo s pomočjo dekodirne matrike $\mathbf{R}$. Iz pomnilnika sprejet dekodirani podatkovni vektor označimo s $\boldsymbol{p_r}$.

$$\boldsymbol{p_r} = \mathbf{R} \cdot \boldsymbol{r} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

Iz pomnilnika sprejet dekodirani podatkovni vektor $\boldsymbol{p_r}$ je enak podatkovnemu vektorju $\boldsymbol{p}$, ki je bil v pomnilnik vpisan.

e) Zdaj predpostavimo, da je prišlo v pomnilniku do enojne napake na petem bitnem mestu – to je na podatkovnem bitu d_2.

$$\boldsymbol{z} = \mathbf{H} \cdot \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \uparrow$$

Najbolj uteženi bit sindroma se nahaja spodaj, zato je njegove bite treba prebrati od spodaj navzgor. Vrednost sindroma je $101_2 = 5$ in s tem pokaže, da je napaka v kodni besedi na bitnem mestu 5. Tam se nahaja podatkovni bit d_2. Tega je treba invertirati in s tem je enobitna napaka popravljena.

f) Zdaj predpostavimo, da je prišlo v pomnilniku do enojne napake na tretjem bitnem mestu – to je na podatkovnem bitu d_1.

$$\boldsymbol{r} = \mathbf{H} \cdot \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \uparrow$$

Vrednost sindroma je $011_2 = 3$ in s tem pokaže, da je napaka v kodni besedi na bitnem mestu 3. Tam se nahaja podatkovni bit d_1. Tega je treba invertirati in s tem je enobitna napaka popravljena.

g) Zdaj predpostavimo, da je prišlo v pomnilniku do enojne napake na drugem bitnem mestu – to je na testnem bitu p_2.

$$\boldsymbol{z} = \mathbf{H} \cdot \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \uparrow$$

Vrednost sindroma je $010_2 = 2$ in s tem pokaže, da je napaka v kodni besedi na bitnem mestu 2. Tam se nahaja testni bit p_2. Ker se nobeden izmed podatkovnih bitov v pomnilniku ni spremenil, ni treba nič popravljati.

h) $$\boldsymbol{z} = \mathbf{H} \cdot \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} \uparrow$$

Vrednost sindroma je $110_2 = 6$, kar naj bi pomenilo, da je prišlo do enojne napake na bitnem mestu 6, torej na podatkovnem bitu d_3. To seveda ni res, saj je prišlo do dveh napak, na podatkovnih bitih d_1 in d_2. Vidimo torej, da Hammingov kod (7,4) ob ugotovljenem sindromu, ki je različen od 0, ne more razlikovati med situacijama, ko sta se spremenila dva bita ali samo eden. Če se je spremenil samo en bit, ga lahko popravi, če pa sta se spremenila dva bita, tega ne more odkriti in zmotno invertira tisti bit, na katerega pokaže izračunani sindrom.

5.13 Popravljanje enojnih/odkrivanje dvojnih napak s kodom (8,4)

V problemu **5.12.h)** smo pokazali, da Hammingov kod (7,4) ne more potrditi, da je v kodni besedi prišlo do dvojne napake. Če pa k štirim podatkovnim bitom dodamo še četrti testni bit p_4, ki se izračuna kot soda pariteta nad celotno kodno besedo, to je nad štirimi podatkovnimi in tremi paritetnimi biti, dobimo Hammingov kod (8,4), ki poveča minimalno Hammingovo razdaljo 3 v kodu (7,4) na 4 v kodu (8,4). Zaradi tega lahko poleg popravljanja enojnih napak odkrije tudi dvojne napake, torej na dveh različnih bitih kodne besede. Vse zmožnosti odkrivanja in popravljanja napak Hammingovega koda (8,4) so zbrane v spodnji tabeli.

Število napak	**Pariteta**	**Sindrom**	**Komentar**
0	soda	0	ni napak
1	liha	0	napaka v p_4
1	liha	$\neq 0$	enojna napaka, nanjo kaže sindrom
2	soda	$\neq 0$	dvojna napaka

a) Za 4-bitno podatkovno besedo $d_1d_2d_3d_4 = 1011$ zapišite 8-bitno kodno besedo v Hammingovem kodu (8,4).

b) Izračunajte sindrom in pariteto, če v pomnilniku na kodni besedi iz točke **a)** ni prišlo do nobene napake. Komentirajte rezultat.

c) Izračunajte sindrom in pariteto, če je prišlo v pomnilniku na kodni besedi iz točke **a)** do napake na paritetnem bitu p_4. Komentirajte rezultat.

d) Izračunajte sindrom in pariteto, če je prišlo v pomnilniku na kodni besedi iz točke **a)** do napake na podatkovnem bitu d_3. Komentirajte rezultat.

e) Izračunajte sindrom in pariteto, če je prišlo v pomnilniku na kodni besedi iz točke **a)** do napake na dveh različnih podatkovnih bitih, na bitu d_1 in bitu d_2. Komentirajte rezultat.

Rešitev:

a) V Hammingovem kodu (7,4) je kodna beseda sestavljena iz štirih podatkovnih in treh testnih bitov v naslednjem vrstnem redu: $p_1p_2d_1p_3d_2d_3d_4$. V Hammingovem kodu (8,4) je tem sedmim bitom na koncu dodan paritetni bit p_4. Celotna kodna beseda za podatkovne bite 1011 je torej 01100110, kjer so podatkovni biti črni, testni biti iz Hammingovega koda (7,4) rdeči, dodani paritetni (testni) bit koda (8,4) pa je zelen.

b) $$\boldsymbol{z} = \mathbf{H} \times \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \oplus \\ 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \oplus \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \underline{0} \\ 0 \end{bmatrix} \uparrow$$

Matriko **H** za Hammingov kot (8,4) dobimo iz matrike za kod (7,4), ki ji na koncu vsake vrstice dodamo paritetni bit p_4 in četrto vrstico s samimi enicami, ker se paritetni bit p_4 računa nad vsemi biti kodne besede. Rezultat je štiribitni stolpec, v katerem spodnji bit pod vodoravno črtico predstavlja pariteto, trije biti nad njim pa v smeri od spodaj navzgor predstavljajo sindrom. Pariteta je soda (0), sindrom ima vrednost $000_2 = 0$, kar ob upoštevanju podane tabele za tolmačenje rezultatov pomeni, da v pomnilniku ni prišlo do nobene napake.

$$\textbf{c)}\ \boldsymbol{z} = \mathbf{H} \times \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \oplus \\ 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \underline{0} \\ 1 \end{bmatrix} \uparrow$$

Vrednost bita p_4, ki je napačen, je označena z rdečo barvo. Izračunana pariteta je liha (1) in sindrom je enak 0. Ta napaka pomeni, da je prišlo do napake na dodanem paritetnem bitu p_4 v kodu (8,4). Ker je pokvarjen paritetni bit, kodne besede ni treba popravljati.

$$\textbf{d)}\ \boldsymbol{z} = \mathbf{H} \times \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \\ 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \underline{1} \\ 1 \end{bmatrix} \uparrow$$

Vrednost bita d_3, ki je napačen, je označena z rdečo barvo. Izračunana pariteta je liha (1) in sindrom je $110_2 = 6$, kar pomeni, da je prišlo do napake na podatkovnem bitu d_3. Prišlo je torej do enojne napake na podatkovnem bitu d_3, ki se lahko invertira in s tem popravi.

$$\textbf{e)}\ \boldsymbol{z} = \mathbf{H} \times \boldsymbol{r} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \times \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \\ 1 \oplus 1 \oplus 1 \oplus 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ \underline{1} \\ 0 \end{bmatrix} \uparrow$$

V pomnilniku je prišlo do dvojne napake – na bitu d_1 na tretjem in bitu d_2 na petem bitnem mestu. Izračunana pariteta je soda in sindrom ima vrednost $110_2 = 6$, kar pomeni, da je prišlo do dvojne napake, torej do dveh napak na različnih bitih. Vidimo, da je Hammingov kod (8,4) sposoben ločiti med dvojno

napako na omenjenih bitih od enojne napake na bitu d_3, česar Hammingov kod (7,4) ne zmore (glejte problem **5.12.h)**).

5.14 Izračun testnih bitov za Hammingov kod (12,8)

Predpostavite, da imamo v pomnilniku shranjeno 8-bitno besedo $d_1d_2d_3d_4d_5d_6d_7d_8 = 01000011$.

a) Določite potrebno število testnih bitov za Hammingov algoritem, ki bo sposoben popravljati enobitne napake.

b) Kakšne so vrednosti testnih bitov, ki se vpišejo k podatkovni besedi v pomnilnik?

c) Napišite razpored bitov in vrednost kodne besede v pomnilniku.

Rešitev:

a) Število podatkovnih bitov $m = 8$. Določiti moramo število potrebnih testnih bitov r. Iz neenačbe $m + r + 1 \leq 2^r$ izhaja, da je najmanjši možni r, ki zadovolji to neenačbo, enak 4. Potrebujemo torej štiri testne bite. Tako dobimo Hammingov kod (12,8).

b) Do enačb za testne bite pridemo, če upoštevamo položaje znakov $\times$ v celotni tabeli pri nalogi **5.11**.

$$\begin{aligned}
p_1 &= d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 &&= 0 \oplus 1 \oplus 0 \oplus 0 \oplus 1 &&= 0\\
p_2 &= d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 &&= 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 &&= 1\\
p_3 &= d_2 \oplus d_3 \oplus d_4 \oplus d_8 &&= 1 \oplus 0 \oplus 0 \oplus 1 &&= 0\\
p_4 &= d_5 \oplus d_6 \oplus d_7 \oplus d_8 &&= 0 \oplus 0 \oplus 1 \oplus 1 &&= 0
\end{aligned}$$

K podatkovni besedi se v pomnilnik vpišejo testni biti $p_1p_2p_3p_4 = 0100$.

c) Razpored in vrednosti bitov kodne besede so prikazani v spodnji tabeli.

Bitno mesto	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Razpored bitov	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8
Vrednosti bitov	0	1	0	0	1	0	0	0	0	0	1	1

5.15 Analiza napake s Hammingovim kodom (12,8)

Za 8-bitno podatkovno besedo $d_1d_2d_3d_4d_5d_6d_7d_8 = 10011100$ bi se pri Hammingovem kodu (12,8) v pomnilnik k njej shranili testni biti $p_1p_2p_3p_4 = 1110$. Ko se celotna kodna beseda prebere iz pomnilnika, se nad njo izračunajo testni biti $p_1'p_2'p_3'p_4' = 1011$. Katera podatkovna beseda je bila prebrana iz pomnilnika [1]?

Rešitev:

Bitno mesto	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**
Razpored bitov	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8
Vpisana beseda	1	1	1	1	0	0	1	0	1	1	0	0
Prebran podatek			1		0	0	1		1	0	0	0

Vrednost celotne kodne besede, ki se vpiše v pomnilnik, je v tabeli prikazana v vrstici z oznako „Vpisana beseda“. Zdaj izračunamo operacijo $\oplus$ nad testnimi biti, ki so bili izračunani in vpisani v pomnilnik ob podatkovnih bitih, in tistimi, ki so bili izračunani, ko je bila beseda prebrana iz pomnilnika: $p_1p_2p_3p_4 \oplus p_1^{'}p_2^{'}p_3^{'}p_4^{'} = 1110 \oplus 1011 = 0101$. Ker je najbolj uteženi testni bit na desni strani, ima rezultat vrednost $1010_2 = 10$. To pomeni, da je napaka v pomnilniku nastopila na bitnem mestu 10 v kodni besedi. Na bitnem mestu 10 se nahaja podatkovni bit d_6. To pomeni, da se je iz pomnilnika prebrala podatkovna beseda $d_1d_2d_3d_4d_5d_6d_7d_8 = 10011000$, kar je razvidno iz vrstice „Prebran podatek“ v tabeli.

5.16 Izračun števila testnih bitov za 1024-bitni podatek

Koliko testnih bitov je potrebnih, če uporabljate Hammingov kod za popravljanje enojnih napak v 1024-bitni podatkovni besedi [1]?

Rešitev:

V neenačbo $2^r - 1 \geq m + r \Rightarrow m + r + 1 \leq 2^r$ vstavimo $m = 1024$ in ugotovimo, da je minimalno število testnih bitov, ki zadovolji to neenačbo, enako 11. Za popravljanje enojnih napak v 1024-bitni podatkovni besedi potrebujemo torej 11 testnih bitov.

5.17 Razvoj Hammingovega koda za 16-bitni podatek*

Razvijte Hammingov kod za 16-bitne podatkovne besede, ki bo znal popravljati enobitne napake.

a) Koliko testnih bitov mora imeti ta kod?

b) Generirajte testne bite za podatkovno besedo $d_1d_2d_3d_4d_5d_6d_7d_8d_9d_{10}d_{11}d_{12}d_{13}d_{14}d_{15}d_{16} = 1101001011001101$.

c) Pokažite, da kod pravilno odkrije napako na podatkovnem bitu d_{13}.

Rešitev:

a) V neenačbo $2^r - 1 \geq m + r \Rightarrow m + r + 1 \leq 2^r$ vstavimo $m = 16$ in ugotovimo, da je minimalno število testnih bitov, ki zadovolji to neenačbo, enako 5. Tako dobimo Hammingov kod (21,5).

b) S povečanjem tabele iz primera **5.11** na 21 bitnih mest in pet testnih bitov lahko pridemo do enačb za posamezne testne bite. Po vstavitvi vrednosti posameznih bitov podane podatkovne besede imajo testni biti naslednje vrednosti:

$$p_1 = d_1 \oplus d_2 \oplus d_4 \oplus d_5 \oplus d_7 \oplus d_9 \oplus d_{11} \oplus d_{12} \oplus d_{14} \oplus d_{16} = 1$$
$$p_2 = d_1 \oplus d_3 \oplus d_4 \oplus d_6 \oplus d_7 \oplus d_{10} \oplus d_{11} \oplus d_{13} \oplus d_{14} = 0$$
$$p_3 = d_2 \oplus d_3 \oplus d_4 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} \oplus d_{15} \oplus d_{16} = 1$$
$$p_4 = d_5 \oplus d_6 \oplus d_7 \oplus d_8 \oplus d_9 \oplus d_{10} \oplus d_{11} = 1$$
$$p_5 = d_{12} \oplus d_{13} \oplus d_{14} \oplus d_{15} \oplus d_{16} = 1$$

Izračunani testni biti, zapisani z najbolj uteženim testnim bitom na levi strani, so: $p_5p_4p_3p_2p_1 = 11101$.

c) Če pride v pomnilniku do napake na podatkovnem bitu d_{13}, se bo spremenila vrednost tistih testnih bitov, ki računajo pariteto tudi nad podatkovnim bitom d_{13}. To sta testna bita p_5 in p_2. Vrednost bita p_5 se iz vrednosti 1 spremeni na 0, vrednost bita p_2 pa iz vrednosti 0 v 1. Testni biti imajo zdaj torej naslednje vrednosti: $p_5p_4p_3p_2p_1 = 01111$. Nad istoležnimi testnimi biti pred in po napaki v pomnilniku izračunamo operacijo XOR:

$$\begin{array}{ll} & 11101 \\ \oplus & \underline{01111} \\ & 10010 = 18 \end{array}$$

Rezultat 10010_2 v desetiškem številskem sistemu pomeni vrednost 18. Na 18. bitnem mestu se nahaja podatkovni bit d_{13}. Tako je odkrita napaka na bitu d_{13}. Napako lahko popravimo, tako da ta bit invertiramo.

6 Zunanji pomnilnik

6.1 Povprečno število prečkanih sledi za dostop na želeno sled na disku*

Zamislite si disk z N sledmi, označenimi od 0 do (N-1), in predpostavite, da so zahtevani sektorji porazdeljeni na disku naključno in enakomerno. Izračunati želimo povprečno število sledi, ki jih je treba prečkati za iskanje sledi, na kateri je zahtevani sektor [1].

a) Najprej izračunajte verjetnost iskanja dolžine j, če se glava trenutno nahaja nad sledjo t. *Namig:* Treba je določiti skupno število kombinacij ob upoštevanju dejstva, da so vsi položaji za cilj iskanja enako verjetni.

b) Zdaj izračunajte verjetnost iskanja dolžine K. *Namig:* Treba je sešteti verjetnosti za vse možne kombinacije premikov prek K sledi.

c) Izračunajte povprečno število prečkanih sledi za iskanje z uporabo formule za pričakovano vrednost

$$E[x] = \sum_{i=0}^{N-1} i \cdot \Pr[x = i] \ .$$

Namig: Uporabite izreka $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ in $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$. Njun dokaz najdete v prilogah **A.3** in **A.4**.

d) Dokažite, da se za velike vrednosti N povprečno število prečkanih sledi pri iskanju približuje $N/3$.

Rešitev:
Najprej si skicirajmo predstavitev N sledi diska:

0	1	$\cdots$	$j-1$	$\cdots$	$N-j$	$\cdots$	$N-2$	$N-1$

a) Uporabimo zapis

$$\mathrm{Ps}[j/t] = \Pr[\text{iskanje dolžine } j, \text{ če je glava trenutno postavljena nad sled } t].$$

Upoštevajmo, da ima vsaka izmed sledi enako verjetnost, da je izbrana. Tako je verjetnost izbire katerekoli sledi $1/N$. Potem lahko zapišemo:

$\mathrm{Ps}[j/t] = 1/N$, če $t \leq j-1$ ali $t \geq N-j$,

$\mathrm{Ps}[j/t] = 2/N$, če $j-1 < t < N-j$.

V prvem primeru je trenutna sled tako blizu enemu robu diska (sledi 0 ali sledi $N-1$), da je samo ena sled od nje oddaljena natanko j sledi. V drugem primeru obstajata dve sledi, ki sta od trenutne sledi t oddaljeni natanko za j sledi (ena levo in druga desno od t), zato je verjetnost iskanja dolžine j verjetnost, da je izbrana katerakoli izmed teh dveh strani, kar znaša $2/N$.

b) Naj bo Ps[K] = Pr[iskanje dolžine K, neodvisno od položaja trenutne sledi]. Dobimo:

$$\mathrm{Ps}\,[K] = \sum_{t=0}^{N-1} \mathrm{Ps}\,[K/t] \cdot \mathrm{Pr}[\text{trenutna sled je sled } t] = \frac{1}{N} \sum_{t=0}^{N-1} \mathrm{Ps}[K/t].$$

Iz točke **a)** vemo, da Ps $[K/t]$ zavzame vrednost $1/N$ za $2K$ sledi in vrednost $2/N$ za (N-$2K$) sledi. Tako dobimo:

$$\mathrm{Ps}\,[K] = \frac{1}{N}\left[\frac{2K}{N} + \frac{2(N-2K)}{N}\right] = \frac{2K + 2(N-2K)}{N^2} = \frac{2}{N} - \frac{2K}{N^2}\ .$$

c)

$$\begin{aligned}
\mathrm{E}\,[K] = \sum_{K=0}^{N-1} K \cdot \mathrm{Pr}\,[K] \ &= \sum_{K=0}^{N-1} \left(\frac{2K}{N} - \frac{2K^2}{N^2}\right) = \frac{2}{N}\sum_{K=0}^{N-1} K - \frac{2}{N^2}\sum_{K=0}^{N-1} K^2 \\
&= \frac{2}{N}\frac{(N-1)\,N}{2} - \frac{2}{N^2}\frac{(N-1)\,N(2N-1)}{6} = (N-1) - \frac{(N-1)\,(2N-1)}{3N} \\
&= \frac{3N\,(N-1) - (N-1)\,(2N-1)}{3N} = \frac{N^2-1}{3N}
\end{aligned}$$

d) Če števec in imenovalec izraza za E[K] v točki **c)** delimo z N, dobimo:

$$\mathrm{E}\,[K] = \frac{N - \frac{1}{N}}{3}\ .$$

Za velike vrednosti N gre člen $1/N$ proti 0 in ga lahko zanemarimo, zato E[K] limitira proti vrednosti $N/3$.

6.2 Povprečni čas dostopa do sektorja na trdem disku

Za diskovni sistem definirajmo naslednje parametre:

T_s = iskalni čas, to je povprečni čas za postavitev glave nad želeno sled,
r = hitrost vrtenja diska v številu obratov na sekundo,
n = število bitov na sektor,
N = kapaciteta sledi v bitih,
T_a = čas dostopa do sektorja.

Izpeljite formulo za T_a kot funkcijo drugih parametrov [1].

Rešitev:
Skupni čas dostopa do sektorja je sestavljen iz treh delov: iz povprečnega iskalnega časa, ki je potreben, da se glava premakne nad sled z želenim sektorjem, iz povprečne vrtilne zakasnitve zaradi čakanja, da pride pod glavo začetek želenega sektorja (v povprečju je to polovica časa za en obrat diska), in samega časa za prenos vseh bitov podatkov v sektorju. Podaja ga spodnja enačba:

$$T_a = T_s + \frac{1}{2r} + \frac{n}{rN}\ .$$

6.3 Izračun parametrov zmogljivosti trdega diska (1/2)

Predpostavite pogon magnetnega diska z osmimi stranmi (površinami) diska, 512 sledmi na eni strani diska in 64 sektorji na sled. Velikost sektorja je 1 KB. Povprečni iskalni čas je 8 ms, čas prehoda s sledi na sosednjo sled je 1,5 ms in disk se vrti s hitrostjo 3.600 rpm (obratov na minuto). Zaporedne sledi v cilindru se lahko berejo brez premika glave [1].

a) Koliko znaša kapaciteta diska?

b) Predpostavite, da je datoteka shranjena v sosednjih sektorjih in sledeh sosednjih cilindrov z začetkom v sektorju 0 sledi 0 cilindra i. Koliko znaša povprečni čas, ki je potreben, da glava prispe do sektorja 0 sledi 0 cilindra i, kjer se začne datoteka?

c) Ocenite čas, ki je potreben za prenos datoteke z dolžino 5 MB.

d) Koliko znaša hitrost prenosa?

Rešitev:

a) Kapaciteta diska $= 8 \cdot 512 \cdot 64 \cdot 1\,\text{KB} = 2^3 \cdot 2^9 \cdot 2^6 \cdot 2^{10}\,\text{B} = 2^{28}\,\text{B} = 256\,\text{MB}$.

b) Vrtilna zakasnitev = (čas za en obhod)/2 = $60/(2 \cdot 3.600) = 8{,}3$ ms.

Povprečni čas dostopa do začetka datoteke = iskalni čas + vrtilna zakasnitev $= 8 + 8{,}3 = 16{,}3$ ms.

c) Vsak cilinder vsebuje 8 sledi · 64 sektorjev/sled · 1 KB/sektor = 512 KB, zato 5 MB podatkov zahteva natanko 10 cilindrov. Disk bo porabil 8 ms, da najde cilinder, v povprečju 8,3 ms, da najde sektor 0, in $8 \cdot (60/3.600) = 133{,}3$ ms, da prebere vseh 8 sledi na enem cilindru. Zdaj se potrebuje 1,5 ms za premik na naslednji sosednji cilinder, kar je ravno čas za premik s sledi na sosednjo sled. Pri prehodu na novi cilinder moramo upoštevati vrtilno zakasnitev. Skupni čas za prenos datoteke je $8 + 8{,}3 + 133{,}3 + 9 \cdot (1{,}5 + 8{,}3 + 133{,}3) = 1.437{,}5$ ms.

d) Hitrost prenosa $= \frac{\text{obrati}}{\text{sekundo}} \cdot \frac{\text{sektorjev}}{\text{obrat}} \cdot \frac{\text{bajtov}}{\text{sektor}} = \frac{3.600}{60} \cdot 64 \cdot 1\,\text{KB} = 3{,}75\,\text{MB/s}$.

6.4 Izračun parametrov zmogljivosti trdega diska (2/2)

Predpostavite disk z eno ploščo in naslednjimi parametri: hitrost vrtenja: 7.200 rpm; število sledi na eni strani plošče: 30.000; število sektorjev na sled: 600; iskalni čas: 1 ms za vsakih 100 prečkanih sledi. Disk je prejel zahtevo za dostop do naključnega sektorja na naključni sledi in predpostavite, da je glava diska v začetku nad sledjo 0 [1].

a) Koliko znaša povprečni iskalni čas?

b) Koliko znaša povprečna vrtilna zakasnitev?

c) Koliko znaša čas prenosa enega sektorja?

d) Koliko znaša skupni povprečni čas za izpolnitev zahteve?

Rešitev:

a) Če predpostavimo, da je glava v začetku postavljena nad sled 0, se izračun poenostavi. Če je zahtevana sled s številko 0, je iskalni čas 0. Če je zahtevana sled s številko 29.999, je iskalni čas tisti čas, ki je potreben za prečkanje 29.999 sledi. Za naključno zahtevo bo povprečno število prečkanih sledi $29.999/2 = 14.999{,}5$ sledi. Pri zakasnitvi 1 ms za 100 sledi je povprečni iskalni čas torej 149,995 ms.

b) Pri hitrosti vrtenja 7.200 rpm naredi disk en obrat v času 8,333 ms. Povprečna vrtilna zakasnitev je tako 4,167 ms.

c) Pri 600 sektorjih na sled in časom za en popoln obrat 8,333 ms je čas prenosa enega sektorja $8{,}333/600 = 0{,}01389$ ms.

d) Skupni povprečni čas za izpolnitev zahteve, torej za dostop do podatkov enega sektorja, je enak vsoti časov pri točkah **a)**, **b)** in **c)**, kar znaša skupaj okrog 154 ms.

6.5 Razlika med fizičnimi in logičnimi zapisi na trdem disku

Navedimo razliko med fizičnimi in logičnimi zapisi. *Logični zapis* je zbirka sorodnih podatkovnih elementov, ki predstavljajo logično celoto, ne glede na to, kako ali kam se informacija shrani. *Fizični zapis* je sosednje področje hranilnega prostora, ki je definiran z značilnostmi hranilne naprave in operacijskega sistema. Predpostavite diskovni sistem, v katerem vsak fizični zapis vsebuje 30 120-bajtnih logičnih zapisov. Izračunajte, koliko prostora na disku (v sektorjih, sledeh in straneh diska) je potrebnega za hrambo 300.000 logičnih zapisov, če ima disk sektorje fiksne kapacitete 512 bajtov, 96 sektorjev na sled, 110 sledi na eni strani plošče in 8 uporabnih strani diska. Zanemarite morebitni(-e) zapis(-e) za glavo datoteke, indekse sledi in predpostavite, da se zapisi ne morejo raztezati prek dveh sektorjev [1].

Rešitev:
Vsak sektor lahko shrani $\lfloor 512/120 \rfloor = 4$ logične zapise. Zahtevano število sektorjev je $300.000/4 = 75.000$ sektorjev. Potrebnih je $\lceil 75.000/96 \rceil = 782$ sledi, kar nadalje pomeni $\lceil 782/110 \rceil = 8$ strani diska.

6.6 Čas prenosa podatkov med sektorji na trdem disku

Imamo disk, ki se vrti s hitrostjo 3.600 rpm. Iskalni čas za premik glave med sosednjima sledema je 2 ms. Disk ima na sledi 32 sektorjev, ki so shranjeni zaporedno od sektorja 0 do sektorja 31. Bralno/pisalna glava dostopa do sektorjev po naraščajočem vrstnem redu. Predpostavite, da je glava na začetku sektorja 1 na sledi 8. V glavnem pomnilniku je dovolj velik vmesnik, da lahko vanj shranimo celotno sled. Podatki med lokacijami na disku se prenašajo z branjem iz izvorne sledi v vmesnik v glavnem pomnilniku in nato pisanjem podatkov iz vmesnika na ciljno sled [1].

a) Koliko časa bi trajal prenos sektorja 1 na sledi 8 na sektor 1 na sledi 9?

b) Koliko časa bi trajal prenos vseh sektorjev sledi 8 na ustrezne sektorje sledi 9?

Rešitev:

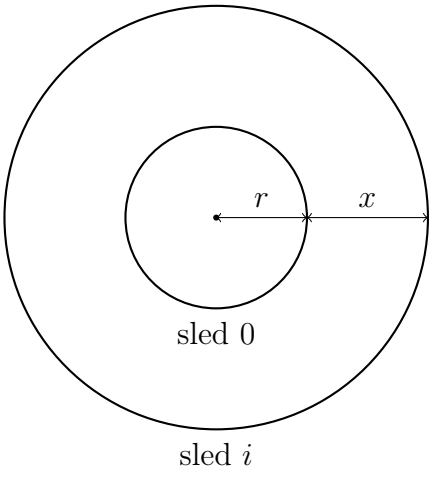
a) Čas je sestavljen iz naslednjih komponent: časa za branje izvornega sektorja, časa za dostop na ciljno sled, vrtilne zakasnitve in časa za pisanje ciljnega sektorja. En obrat diska za branje ali pisanje celotne sledi traja 60/3.600 = 16,7 ms. Čas za branje ali pisanje enega samega sektorja je 16,7/32 = 0,52 ms. Čas za premik na sosednjo sled (iz sledi 8 na sled 9) je 2 ms. Vrtilna zakasnitev je čas, ki se porabi, da se glava spet poravna s sektorjem 1. Ker je po zaključenem branju sektorja glava na njegovem koncu, se mora začetek tega sektorja zavrteti še 31/32 dolžine sledi, da se bo začetek sektorja prvič spet poravnal z glavo. To se zgodi po preteku $16{,}7 \cdot (31/32) = 16{,}2$ ms. Čas, potreben za premik glave s sledi 8 na sled 9, znaša 2 ms in se prekriva s časom 16,2 ms za vrtilno zakasnitev, zato se upošteva samo daljši čas, torej čas vrtilne zakasnitve. Skupni čas prenosa je 0,52 + 16,2 + 0,52 = 17,24 ms.

b) Čas za branje ali pisanje celotne sledi je preprosto čas za en obrat plošče, to je 16,7 ms. Med branjem in pisanjem se porabita 2 ms za premik glave s sledi 8 na sled 9. V tem času so se trije sektorji in večina četrtega že zavrteli mimo glave. Ker pa je v vmesniku shranjena celotna sled, lahko pišemo sektorje na ciljno sled po drugem vrstnem redu, kot smo jih brali z izvorne sledi. Tako se lahko pisanje začne s petim sektorjem (ima številko 4) na sled 9. Ta sektor glava doseže po preteku $0{,}52 \cdot 4 = 2{,}08$ ms po končanju operacije branja. Skupni čas prenosa znaša torej 16,7 + 2,08 + 16,7 = 35,48 ms.

6.7 Izpeljava izraza za gostoto zapisa podatkov

Izpeljite izraz za gostoto zapisa podatkov (d_i) na sledi i trdega diska v odvisnosti od polmera (r) in gostote zapisa podatkov (d) skrajno notranje sledi s številko 0. Predpostavite, da ima trdi disk gostoto D sledi na cm.

Rešitev:



Na zgornji sliki je narisana najbolj notranja sled trdega diska s polmerom r in sled i, ki ima polmer $r + x$. Količina podatkov na sledi (v B) je enaka dolžini sledi oz. obsegu krožnice (v cm), pomnoženi z gostoto zapisa podatkov na sledi (v B/cm).

Ker je na vsaki sledi diska enaka količina podatkov, lahko zapišemo, da je količina podatkov na skrajno notranji sledi s številko 0 enaka količini podatkov na sledi i:

$$2\pi rd = 2\pi(r+x)d_i \ .$$

Na razdalji x se nahaja i sledi z gostoto D sledi na cm, zato lahko razdaljo x zapišemo kot

$$x = \frac{i}{D} \ .$$

Izraz za x vstavimo v prvo enačbo in iz nje izrazimo gostoto zapisa podatkov na sledi i:

$$d_i = \frac{rd}{r + \frac{i}{D}} \ .$$

6.8 Redundantno polje neodvisnih diskov

Imamo redundantno polje neodvisnih diskov (RAID) s štirimi diski. Vsak ima kapaciteto 200 GB (gigabajtov). Koliko znaša razpoložljiva kapaciteta hrambe podatkov za vsakega izmed naslednjih nivojev RAID: 0, 1, 3, 4, 5, 6 [1]?

Rešitev:
Označimo z N potrebno število podatkovnih diskov za vsak nivo RAID.

Nivo RAID	**Izračun števila podatkovnih diskov**	**Razpoložljiva kapaciteta hrambe podatkov (GB)**
RAID 0	$N = 4$	800
RAID 1	$2 \cdot N = 4 \Rightarrow N = 2$	400
RAID 3	$N + 1 = 4 \Rightarrow N = 3$	600
RAID 4	$N + 1 = 4 \Rightarrow N = 3$	600
RAID 5	$N + 1 = 4 \Rightarrow N = 3$	600
RAID 6	$N + 2 = 4 \Rightarrow N = 2$	400

6.9 Varnostno shranjevanje podatkov

Razvijte strategijo za varnostno shranjevanje podatkov v računalniškem sistemu. Ena izbirna možnost je uporabiti priklop zunanjih diskov, kar stane 150 USD za vsak pogon 500 GB. Druga izbirna možnost je kupiti tračni pogon za 2.500 USD in tračne kasete s kapaciteto 400 GB po 50 USD. Tipična strategija varnostnega shranjevanja podatkov je imeti dve množici medijev za hrambo na lokaciji računalnika in varnostne kopije shranjevati nanju izmenično, tako da je v primeru, če se sistem zruši v času kopiranja, prejšnja verzija še vedno nepoškodovana. Obstaja še tretja množica medijev na drugi lokaciji, ki se periodično ažurira s podatki z množice medijev na sami lokaciji [1].

a) Predpostavite, da morate varnostno shraniti 1 TB (terabajtov) podatkov. Koliko bi stal diskovni sistem za varnostno shranjevanje podatkov?

b) Koliko bi stal tračni sistem za varnostno shranjevanje 1 TB podatkov?

c) Pri kako veliki količini podatkov za vsako varnostno shranjevanje bi bila strategija shranjevanja na trak cenejša?

d) Katera vrsta strategije za varnostno shranjevanje daje prednost traku?

Rešitev:

a) Stroški diskovnega sistema za varnostno shranjevanje 1 TB podatkov bi znašali $2 \cdot 3 \cdot 150 = 900\,\text{USD}$.

b) Stroški tračnega sistema za varnostno shranjevanje 1 TB podatkov bi znašali $2.500 + 3 \cdot 3 \cdot 50 = 2.950\,\text{USD}$.

c) Naj bo Z število GB, pri katerih bi nas oba pristopa za varnostno shranjevanje podatkov stala približno enako. Cena za sistem varnostnega shranjevanja z diski je $C_d = (Z/500) \cdot 3 \cdot 150\,\text{USD}$. Cena za sistem varnostnega shranjevanja s tračnim pogonom je $C_t = 2.500 + (Z/400) \cdot 3 \cdot 50\,\text{USD}$. Pri enakih stroških za oba načina varnostnega shranjevanja podatkov velja $C_d = C_t$. Iz enačbe izrazimo Z in dobimo $Z = 4.762\,\text{GB}$. Če moramo varnostno shranjevati več kot 4.762 GB podatkov, je strategija shranjevanja na trak cenejša.

d) Shranjevanje na trak bi bilo bolj primerno, če bi želeli imeti več kopij vsakokratnega varnostnega shranjevanja podatkov.

6.10 Serpentinasto zapisovanje podatkov na magnetni trak

Magnetni trak je dolg 300 m in širok 12 mm. Podatki se zapisujejo z 32 sledmi v tehniki serpentinastega zapisovanja, kar pomeni, da se zapis vrši izmenično v eno in drugo smer. Gostota zapisa na sled po dolžini traku znaša 62,5 kB/m. Trak se premika s hitrostjo 1,5 m/s. *Opomba:* Pri tej nalogi upoštevajte, da 1 kB pomeni 1.000 B, saj se v tehničnih specifikacijah za shranjevanje podatkov (trakovi, optični diski in trdi diski) praviloma uporablja desetiški številski sistem.

a) Izračunajte skupno količino podatkov, ki se lahko zapiše na trak.

b) Izračunajte čas, ki je potreben za popoln zapis vseh podatkov.

Rešitev:

a) Skupno količino podatkov, ki se lahko zapiše na trak, izračunamo po naslednjih korakih.

Število prehodov (naprej-nazaj) je enako številu sledi, torej 32.
Dolžina zapisa na vseh sledeh skupaj $L = 32 \cdot 300\,\text{m} = 9.600\,\text{m}$.
Gostota zapisa $d = 62{,}5\,\text{kB/m} = 62{,}5 \cdot 10^3\,\text{B/m}$.
Skupna količina podatkov, ki se lahko zapiše na trak, je
$L \cdot d = 9.600\,\text{m} \cdot 62.500\,\text{B/m} = 600.000.000\,\text{B} = 600\,\text{MB}$.

b) Čas zapisovanja, ki je potreben za popoln zapis vseh podatkov, je

$$t = \frac{9.600\,\text{m}}{1{,}5\,\text{m/s}} = 6.400\,\text{s} \approx 1\,\text{h}\,46\,\text{min}\,40\,\text{s}.$$

6.11 Izračun števila sektorjev na DVD ROM-u

Koliko sektorjev po 2.048 bajtov vsebuje enoslojni DVD-ROM s kapaciteto 4,7 GB? *Opomba:* Pri tej nalogi upoštevajte, da 1 GB pomeni 10^9 B, saj se v tehničnih specifikacijah za shranjevanje podatkov (trakovi, optični diski in trdi diski) praviloma uporablja desetiški sistem.

Rešitev:
Kapaciteto pretvorimo v bajte: $4{,}7\,\mathrm{GB} = 4{,}7 \cdot 10^9\,\mathrm{B}$. Dobimo

$$\frac{4{,}7 \cdot 10^9\,\mathrm{B}}{2.048\,\mathrm{B}} = 2.294.921{,}875.$$

Ker je število sektorjev lahko samo celo število, upoštevamo, da enoslojni DVD-ROM s kapaciteto 4,7 GB vsebuje 2.294.921 sektorjev po 2.048 bajtov.

6.12 Hitrost vrtenja CD-ROM-a

Optični senzor na CD-ROM-u bere podatke na notranjem robu diska pri polmeru $r = 25\,\mathrm{mm}$. Hitrost prenosa podatkov je $1\times$ (tj. $150\,\mathrm{kB/s} = 150.000\,\mathrm{B/s}$). Gostota podatkov vzdolž spirale (število bitov na centimeter poti bralne glave) znaša $D = 10\,\mathrm{kb/cm} = 10.000\,\mathrm{b/cm}$. Izračunajte, s kakšno hitrostjo (v obratih na minuto, rpm) se mora vrteti disk. *Opomba:* V tej nalogi je hitrost 150 kB/s zapisana v desetiškem številskem sistemu, kar pomeni $150\,\mathrm{kB/s} = 150.000\,\mathrm{B/s}$. To sledi industrijskim standardom za optične nosilce podatkov, kjer je $1\,\mathrm{kB} = 1.000\,\mathrm{B}$.

Rešitev:
Pretvorba enote za hitrost prenosa: $150\,\mathrm{kB/s} = 150.000\,\mathrm{B/s} = 1.200.000\,\mathrm{b/s}$.

Polmer: $r = 25\,\mathrm{mm} = 2{,}5\,\mathrm{cm}$.

Gostota podatkov: $D = 10.000\,\mathrm{b/cm}$.

Tangencialna (linearna) hitrost branja:

$$\nu = \frac{\text{bitna hitrost}}{D} = \frac{1.200.000\,\mathrm{b/s}}{10.000\,\mathrm{b/cm}} = 120\,\mathrm{cm/s} = 1{,}2\,\mathrm{m/s}.$$

Vrtilna frekvenca:

$$f = \frac{v}{2\pi r} = \frac{120\,\mathrm{cm/s}}{2\pi \cdot 2{,}5\,\mathrm{cm}} = \frac{120}{5\pi}\,\mathrm{s}^{-1} \approx 7{,}639\,\mathrm{Hz}.$$

Hitrost vrtenja v obratih na minuto: $60 \cdot f \approx 60 \cdot 7{,}639 \approx 458{,}3\,\mathrm{rpm}$.

Disk se mora vrteti približno s hitrostjo $4{,}58 \cdot 10^2\,\mathrm{rpm}$.

6.13 Prilagajanje kotne hitrosti glede na položaj senzorja

Optični senzor bere podatke na dveh različnih razdaljah od središča diska: najprej pri polmeru $r_1 = 30\,\mathrm{mm}$, nato pa pri $r_2 = 60\,\mathrm{mm}$. Ko je senzor pri r_1, se disk vrti s kotno hitrostjo $\omega_1 = 120\,\mathrm{rad/s}$.

a) Izračunajte linearno hitrost premikanja diska mimo optičnega senzorja pri prvem položaju.

b) Kolikšna mora biti nova kotna hitrost ω_2, da bo linearna hitrost premikanja diska mimo optičnega senzorja ostala enaka tudi pri polmeru r_2?

Rešitev:

a) Linearna hitrost premikanja diska mimo optičnega senzorja pri polmeru r_1 je

$$\nu_1 = \omega_1 \cdot r_1 = 120 \cdot 0{,}03 = 3{,}6\,\mathrm{m/s}$$

b) Zaradi zahteve po konstantni linearni hitrosti (hitrosti gibanje diska mimo bralnega laserskega žarka) se mora disk za dostope bližje zunanjemu robu (npr. pri polmeru r_2) vrteti počasneje kot za dostope bližje notranjemu robu (npr. pri polmeru r_1), torej $\omega_1 \geq \omega_2$. Veljati mora torej $\omega_1 r_1 = \omega_2 r_2$. Kotna hitrost ω_2 pri polmeru r_2 je zato

$$\omega_2 = \frac{r_1}{r_2} \cdot \omega_1 = \frac{30\,\mathrm{mm}}{60\,\mathrm{mm}} \cdot 120\,\mathrm{rad/s} = 60\,\mathrm{rad/s}$$

Da ostane linearna hitrost enaka pri $r_2 = 60\,\mathrm{mm}$, mora biti tam kotna hitrost $\omega_2 = 60\,\mathrm{rad/s}$.

7 Vhodno-izhodna organizacija mikroprocesorjev

7.1 Naslavljanje vhodno-izhodnih vrat pri mikroprocesorju Intel 8088

V tipičnem mikroprocesorju se v vhodno-izhodnem (V/I) krmilniku za dano napravo uporablja ločen V/I naslov za sklicevanje na V/I podatkovna registra (podatkovni vhodni register in podatkovni izhodni register) in ločen V/I naslov za en krmilni in en statusni register. Takim registrom pravimo vrata (port). Pri mikroprocesorju Intel 8088 se uporabljata dva formata V/I ukazov. Pri enem formatu 8-bitna operacijska koda specificira V/I operacijo, sledi pa ji 8-bitni naslov V/I vrat. Druge operacijske kode za V/I vsebujejo naslov V/I vrat v 16-bitnem registru DX. Koliko vrat lahko mikroprocesor 8088 naslovi v vsakem od teh načinov naslavljanja [1]?

Rešitev:
V prvem načinu naslavljanja lahko procesor naslovi $2^8 = 256$ V/I vrat. Tipično bi to omogočalo naslovitev 128 naprav. Vendar pa operacijska koda specificira bodisi vhodno ali izhodno operacijo, zato je naslove mogoče pouporabiti, tako da imamo 256 naslovov vhodnih vrat in 256 naslovov izhodnih vrat, kar omogoča naslovitev 256 naprav. V drugem načinu naslavljanja je mogočih $2^{16} = 64$K naslovov vrat.

7.2 Naslavljanje vhodno-izhodnih vrat pri mikroprocesorju Zilog Z8000

V družini mikroprocesorjev Zilog Z8000 obstaja zmožnost neposrednega naslavljanja vrat, pri katerem je 16-bitni naslov vrat del samega ukaza, in zmožnost posrednega naslavljanja vrat, pri katerem se ukaz sklicuje na enega izmed 16-bitnih splošnonamenskih registrov, ki vsebujejo naslov vrat. Koliko vrat lahko naslovi Z8000 v vsakem V/I načinu naslavljanja [1]?

Rešitev:
Z neposrednim načinom naslavljanja lahko ukaz naslovi do $2^{16} = 64$K vrat. Pri posrednem načinu naslavljanja je naslov vrat v 16-bitnih registrih, zato lahko tudi pri tem načinu naslavljanja ukaz naslovi do $2^{16} = 64$K vrat.

7.3 Blokovni V/I prenosi mikroprocesorja Zilog Z8000

Mikroprocesor Zilog Z8000 vključuje tudi zmožnost blokovnega V/I prenosa, ki je za razliko od DMA-ja pod neposrednim nadzorom procesorja. Ukazi za prenos blokov specificirajo register za naslov vrat (Rp), števni register (Rc) in ciljni register (Rd). Rd vsebuje naslov v glavnem pomnilniku, na katerega se shrani prvi prebrani bajt iz vhodnih vrat. Rc je katerikoli izmed 16-bitnih splošnonamenskih registrov. Kako veliki podatkovni bloki se lahko prenašajo [1]?

Rešitev:
Ker je Rc 16-bitni register, se lahko prenašajo bloki velikosti $2^{16} = 64$ KB.

7.4 Blokovni in neblokovni V/I prenosi za Zilog Z8000

Zamislimo si mikroprocesor, kot je Zilog Z8000, ki ima ukaz za blokovni V/I prenos. Po prvi izvedbi ukaza traja vsaka ponovna izvedba pet urinih ciklov. Če pa uporabimo ukaz za neblokovni V/I prenos, znaša skupni čas za včitanje in izvajanje ukaza 20 urinih ciklov. Izračunajte pohitritev prenosa z ukazom za blokovni prenos med prenosom blokov velikosti 128 bajtov [3].

Rešitev:
Z uporabo neblokovnih V/I ukazov traja prenos $20 \cdot 128 = 2.560$ urinih ciklov. Z blokovnim V/I traja prenos $5 \cdot 128 = 640$ urinih ciklov (ob zanemaritvi časa za enkratno včitanje iterativnega ukaza in njegovih operandov). Pohitritev prenosa znaša $(2.560 - 640)/2.560 = 0{,}75$ ali 75 %.

7.5 Registri in naslovi V/I naprave

Nek sistem temelji na 8-bitnem mikroprocesorju in ima dve V/I napravi. V/I krmilnik za ta sistem uporablja ločene krmilne in statusne registre. Obe napravi obravnavata podatke po en bajt naenkrat. Prva naprava ima dve statusni liniji in tri krmilne linije. Druga naprava ima tri statusne linije in štiri krmilne linije [3].

a) Koliko 8-bitnih registrov V/I krmilnega modula potrebujemo za branje statusa in krmiljenje obeh naprav?

b) Kolikšno je skupno število potrebnih registrov krmilnega modula, če je prva naprava samo izhodna?

c) Koliko različnih naslovov je potrebnih za krmiljenje obeh naprav?

Rešitev:

a) Vsaka V/I naprava potrebuje ena izhodna vrata za krmilne ukaze in ena vhodna vrata za status. Širina vsakih vrat je določena z ustreznim številom statusnih in krmilnih linij. Na primer, statusna vrata prve naprave potrebujejo le dva bita.

b) Prva naprava potrebuje samo ena vrata za podatke, medtem ko druga naprava potrebuje tako vhodna kot tudi izhodna podatkovna vrata. Ker vsaka naprava potrebuje ena krmilna in ena statusna vrata, je skupaj potrebnih sedem vrat.

c) Za krmiljenje obeh naprav potrebujemo sedem različnih naslovov. Če pa upoštevamo, da lahko uporabimo isti naslov za dostop do dvojih vrat, če so ena vhodna, druga pa izhodna, lahko shajamo samo s štirimi različnimi naslovi.

7.6 Periodično preverjanje statusa naprave

Pri programskem V/I procesor obtiči v čakalni zanki za preverjanje statusa V/I naprave. Da bi povečali učinkovitost, bi lahko programsko opremo za V/I napisali tako, da procesor periodično preverja status naprave. Če naprava ni pripravljena, lahko procesor skoči na izvajanje drugih opravil. Po preteku nekega časovnega intervala se procesor vrne na ponovno preverjanje statusa [1].

a) Predpostavite zgornjo shemo za oddajo podatkov znak za znakom na tiskalnik, ki lahko dela s hitrostjo 10 znakov na sekundo (cps). Kaj se bo zgodilo, če se njegov status preverja na vsakih 200 ms?

b) Zdaj predpostavite tipkovnico z vmesnikom za en sam znak. V povprečju se znaki vnašajo s hitrostjo 10 cps, vendar lahko časovni interval med dvema zaporednima pritiskoma tipke znaša zgolj 60 ms. S kakšno frekvenco naj bi V/I program pregledoval tipkovnico?

Rešitev:

a) Če preverjamo status tiskalnika samo na vsakih 200 ms, se hitrost tiskanja upočasni na okrog 5 cps.

b) Vhodne naprave moramo obravnavati drugače, da se vhodni podatek ne izgubi. Vmesnik je treba preverjati vsaj na vsakih 60 ms. V nasprotnem primeru tvegamo, da nazadnje vtipkani znak v vmesniku prepiše prejšnjega, ki še ni bil prebran. Tako lahko pride do izgubljanja nekaterih znakov.

7.7 Strežba naprave s programskim V/I

Nek mikroprocesor preverja status V/I naprave vsakih 20 ms, kar je izvedeno s časovnikom, ki procesor alarmira o vsakokratnem preteku tega časovnega intervala. Vmesnik naprave vsebuje dvoje vrat: ena za status in ena za podatkovni izhod. Koliko časa traja preverjanje in strežba naprave, če je frekvenca ure 8 MHz? Zaradi enostavnosti predpostavite, da vsi ukazni cikli trajajo 12 urinih ciklov [3].

Rešitev:
Pri frekvenci ure 8 MHz znaša perioda ure 0,125 μs, zato ukazni cikel traja $12 \cdot 0{,}125 = 1{,}5\,\mu$s. Za preverjanje statusa potrebujemo en ukaz vhodnega tipa za branje statusnega registra naprave, nato pa še vsaj en ukaz za pregled vsebine registra. Če je naprava pripravljena, potrebujemo en ukaz izhodnega tipa za prenos podatka k napravi. Skupno so torej potrebni trije ukazi, kar zahteva $3 \cdot 1{,}5 = 4{,}5\,\mu$s.

7.8 Prednosti in slabosti pomnilniško preslikanega V/I

Vhodno/izhodne naprave v mikroračunalnikih lahko uporabljajo bodisi pomnilniško preslikan V/I ali izoliran V/I [1].

a) Navedite prednosti pomnilniško preslikanega V/I.

b) Navedite slabosti pomnilniško preslikanega V/I.

Rešitev:

a) Prednosti pomnilniško preslikanega V/I so:

- Za delo z V/I vrati lahko uporabljamo velik nabor ukazov, kar dopušča bolj učinkovito programiranje.
- Ne potrebujemo nobene dodatne krmilne linije na vodilu, da bi razlikovali med dostopom do pomnilnika in dostopom do V/I prostora.

- Naslavljanje je prožnejše. Na primer, v naboru ukazov se lahko uporabljajo različni načini naslavljanja in različni registri za izmenjavo podatkov z V/I moduli.

b) Slabosti pomnilniško preslikanega V/I so:

- Za naslavljanje V/I naprav se porablja dragocen pomnilniški naslovni prostor.
- Pomnilniško preslikan V/I uporablja ukaze za dostop do pomnilnika, ki so pri večini procesorjev daljši od V/I ukazov. Dolžina programov je zaradi tega večja.
- Strojna oprema naslovnih dekodirnikov za V/I module je kompleksnejša, ker je naslov naprave daljši.

7.9 Ločen V/I v primerjavi s pomnilniško preslikanim V/I

Včasih so V/I naprave opremljene z vmesnimi pomnilniki, ki sprejmejo bloke podatkov. Tak vmesni pomnilnik lahko CPE napolni (ali izprazni) s podatki iz lastnega pomnilnika. Iterativni tipi ukazov so zelo učinkoviti pri pospeševanju takih prenosov. Težava je v tem, da veliko CPE nima iterativnih V/I ukazov. Na primer, mikroprocesor Intel 8086 ima iterativne ukaze samo za prenose pomnilnik-pomnilnik. Če pa uporabimo pomnilniško preslikan V/I, jih lahko izkoristimo [3].

a) Zamislite si sistem na osnovi mikroprocesorja Intel 8086, ki uporablja ločen V/I naslovni prostor za prenose podatkov k oziroma od naprave, opremljene z vmesnim pomnilnikom kapacitete 128 bajtov. Podatkovni blok se prenese z izvajanjem programske zanke z uporabo „vhodnih“ ali „izhodnih“ ukazov mikroprocesorja 8086. Programska zanka prenese eno 16-bitno besedo na iteracijo in za izvajanje porabi 50 urinih ciklov. Izračunajte, koliko traja prenos bloka podatkov dolžine 128 bajtov pri frekvenci ure mikroprocesorja 5 MHz.

b) Predpostavimo, da namesto tega uporabimo pomnilniško preslikan V/I. Zdaj imamo možnost uporabe ukaza „move string“, ki traja samo 17 urinih ciklov na besedo. Izračunajte faktor, za katerega se pospeši prenos podatkov (v odstotkih).

c) Vmesni pomnilnik V/I naprave se napolni z novimi podatki dvestokrat na sekundo. Kolikšen delež CPE časa se porabi za prenose na napravo oziroma iz naprave? Izračunajte ta deleža za sistema, obravnavana v točkah **a)** in **b)**.

d) Vmesni pomnilnik V/I naprave se napolni enkrat na sekundo. Ali je res pomembno, ali uporabimo programsko zanko ali ukaz za procesiranje nizov?

Rešitev:

a) Prenos bloka dolžine 128 bajtov traja $(128/2) \cdot 50 = 3.200$ urinih ciklov in zato $3.200 \cdot 0{,}2 = 640\,\mu$s.

b) Prenos bloka z ukazom „move string“ traja $64 \cdot 17 \cdot 0{,}2 = 218\,\mu$s. Hitrost prenosa bloka podatkov se je v primerjavi s hitrostjo, doseženo v točki **a)**, povečala za $((640 - 218)/640) \cdot 100\ \%) = 66\ \%$.

c) V sistemu iz točke **a)** CPE za prenose porabi $200 \cdot 640 \cdot 10^{-6}\,\mathrm{s} = 0{,}13\,\mathrm{s}$ na sekundo, kar pomeni 13 % svojega časa. V sistemu iz točke **b)** CPE za prenose porabi samo $200 \cdot 218 \cdot 10^{-6}\,\mathrm{s} = 0{,}044\,\mathrm{s}$ na sekundo oziroma 4,4 % svojega časa.

d) Za tako redke prenose podatkov je vpliv na porabljen čas CPE zanemarljiv ne glede na to, kateri izmed dveh pristopov se uporablja.

7.10 Prekinitveno voden V/I

Zamislite si sistem z uporabo prekinitveno vodenega V/I za neko napravo, ki prenaša podatke s stalno povprečno hitrostjo 8 kB/s. Takšna „naprava“ bi lahko bila na primer telekomunikacijska linija [1].

a) Predpostavite, da procesiranje prekinitve traja okrog $100\,\mu$s (to je čas za skok v prekinitveni program, njegovo izvedbo in povratek v glavni program). Izračunajte delež procesorskega časa, ki se porabi za to V/I napravo, če ta prekinja za vsak bajt.

b) Zdaj predpostavite, da ima naprava dva 16-bajtna vmesnika in prekinja procesor, ko se napolni eden izmed njiju. Seveda procesiranje prekinitve zdaj traja dlje, ker mora prekinitveni program prenesti 16 bajtov. V prekinitvenem programu traja prenos vsakega bajta $8\,\mu$s. Izračunajte delež procesorskega časa, ki se v tem primeru porabi za to V/I napravo.

c) Nazadnje predpostavite, da procesor premore V/I ukaz za blokovni prenos, kot ga ima mikroprocesor Zilog Z8000. Ta pridruženemu prekinitvenemu programu omogoča prenos vsakega bajta v samo $2\,\mu$s. Izračunajte delež procesorskega časa, ki se tokrat porabi za to V/I napravo.

Rešitev:

a) V povprečju naprava prenese en bajt v času $\frac{1}{8 \cdot 10^3} = 125\,\mu$s. CPE porabi $100\,\mu$s za procesiranje pridružene prekinitve – to je $\frac{100}{125} = 0{,}8$ ali 80 % svojega časa. Zaradi strežbe V/I naprave ostane CPE le malo časa za druga opravila.

b) V tem primeru je časovni interval med prekinitvami $16 \cdot 125 = 2.000\,\mu$s. Vsaka prekinitev zdaj zahteva $100\,\mu$s za prvi znak in čas za prenos vseh preostalih znakov, kar znese skupaj $100 + 15 \cdot 8 = 220\,\mu$s. Delež porabljenega procesorskega časa za to napravo je $\frac{220}{2.000} = 0{,}11$ ali 11 %.

c) V točki **b)** je prekinitveni program prenašal vsak bajt bloka z izvajanjem programske zanke, sestavljene iz več ukazov. Ti ukazi so prenašali vsak bajt v register CPE in od tod v pomnilnik ali V/I napravo (odvisno od smeri prenosa). Vsak ukaz v zanki se mora pri vsakem obhodu zanke vsakokrat včitati, dekodirati in izvesti. Ukaz za blokovni prenos odpravi potrebo po taki programski zanki. Zdaj prekinitveni program včita ukaz samo enkrat in ga ponavljajoče izvaja, dokler ni prenesenih vseh 16 bajtov. Zaradi tega se zmanjša čas procesiranja prekinitve za $16 \cdot 6 = 96\,\mu$s. Porabljen del procesorskega časa v intervalu 2 ms je $\frac{0{,}22 - 0{,}096}{2} = 0{,}062$, to je 6,2 %. Faktor izboljšanja znaša skoraj 2.

7.11 Zamenjava programskega V/I s prekinitveno vodenim

Nek sistem nadzira operater z ukazi, ki jih vnaša prek tipkovnice. Povprečno število ukazov, vnesenih v osemurnem delovniku, je 60. Tipični ukaz vsebuje okrog 48 znakov [3].

a) Predpostavite, da procesor preverja tipkovnico vsakih 100 ms. Kolikokrat bo tipkovnica preverjana v časovnem intervalu osmih ur?

b) Za koliko odstotkov bi se zmanjšalo število procesorjevih preverjanj tipkovnice, če bi uporabili prekinitveno voden V/I?

Rešitev:

a) Pri programskem V/I procesor preverja tipkovnico 10-krat na sekundo. Število pregledov v osmih urah je $10 \cdot 60 \cdot 60 \cdot 8 = 288.000$.

b) Pri prekinitveno vodenem V/I tipkovnica prekine CPE, kadarkoli pritisnemo tipko. CPE streže tipkovnico samo ob nastopu prekinitev. Skupno število znakov, ki jih vtipka operater v osmih urah, je samo $60 \cdot 48 = 2.880$. Število dostopov CPE do tipkovnice se torej zmanjša za $\frac{288.000-2.880}{288.000} = 0{,}99$, to je za 99 %.

7.12 Prekinitve s programskim izpraševanjem

Nekatere CPE niso sposobne obravnavati vektoriziranih prekinitev. Ko se aktivira edina linija za prekinitveno zahtevo takšne CPE, se izvede skok na fiksni pomnilniški naslov, kjer se nahaja skupni prekinitveni program. Ta prebere status V/I naprave z najvišjo prioriteto, da ugotovi, ali je to naprava, ki zahteva strežbo. Če ni tako, prekinitveni program nadaljuje s pregledom statusa naprave z neposredno nižjo prioriteto in tako naprej, dokler ne odkrije izvora prekinitvene zahteve. Nato CPE skoči v drugi program, ki streže tej napravi. Takšna prekinitvena shema se imenuje *programsko izpraševanje*. Za razliko od vektorskih prekinitev ne zahteva zunanjih prekinitvenih vezij. Vsak prekinitveni vir preprosto aktivira skupno linijo za prekinitveno zahtevo na CPE prek vezja z odprtim kolektorjem. Po drugi strani pa povečuje režijo procesiranja prekinitve in upočasni odziv CPE na prekinitve [3].

a) Predstavljajte si sistem, ki uporablja prekinitve s programskim izpraševanjem in ima skupno osem prekinitvenih naprav. Po skoku v skupni prekinitveni program CPE porabi 10 μs za branje in pregled statusa vsake naprave. Koliko časa se porabi za izpraševanje, ko prekinitev povzroči naprava z najnižjo prioriteto? Predpostavite, da ni čakajočih prekinitev.

b) Predpostavite, da je hitrost, s katero se generirajo prekinitve, enaka za vseh osem naprav. Kolikšen je povprečni čas, ki se porabi za programsko izpraševanje naprave? Spet predpostavite, da ni čakajočih prekinitev.

Rešitev:

a) Pred preverjanjem statusa naprave z najnižjo prioriteto to stori CPE za vseh sedem drugih naprav. Za to porabi $7 \cdot 10 = 70\,\mu$s.

b) Verjetnost, da prihaja prekinitev od 1. naprave, je $1/8 = 0{,}125$, da prihaja od 2. naprave prav tako 0,125 in tako naprej. V povprečju je čas, ki se porabi za programsko izpraševanje naprave $0{,}125 \cdot (1+2+3+4+5+6+7) \cdot 10\,\mu\text{s} = 35\,\mu\text{s}$.

7.13 Odzivni čas na prekinitev

Načrtovalec želi ugotoviti, koliko časa je potrebnega do vstopa v prekinitveni program za napravo z najvišjo prioriteto. Iz uporabniškega priročnika za CPE in razgovorov s programskimi inženirji je načrtovalec zbral podatke, zbrane v spodnji tabeli.

Opravilo	**Cikli**
dovršitev najdaljšega ukaza	50
aktiviranje vhoda za prekinitev na CPE po zahtevi V/I naprave	2
potrditev prekinitve in branje prekinitvenega vektorja od V/I naprave	16
shranjevanje vsebin registrov CPE na sklad	60
dostop do začetka prekinitvenega programa s prek. vektorjem	52

Ti podatki podajajo trajanja različnih opravil, ki zadevajo procesiranje prekinitve, izražena v številu urinih ciklov. Na podlagi teh informacij in dejstva, da CPE uporablja uro frekvence 5 MHz, odgovorite na naslednja vprašanja [3].

a) Izračunajte odzivni čas na prekinitveno zahtevo najvišje prioritete za najslabši primer (ignorirajte morebitne čakajoče prekinitve; odzivni čas je čas, potreben za dostop do prekinitvenega programa).

b) V katerem razponu bodo variirali odzivni časi za prekinitve najvišje prioritete?

c) Verjetnosti, da trenutni ukazni cikel za zaključitev potrebuje 20, 10 ali 6 urinih ciklov (ko se pojavi prekinitev), so 20, 30 in 50 %. Izračunajte povprečni čas odziva na prekinitve.

Rešitev:

a) Do najslabšega primera pride tedaj, kadar CPE ravno začne izvajati najdaljši ukazni cikel, ko se aktivira njegov vhod za zahtevo po prekinitvi. Takrat je odzivni čas $(2 + 50 + 16 + 60 + 52) \cdot 0{,}2 = 36\,\mu$s.

b) Odzivni čas bo najkrajši, ko se prekinitveni vhod na CPE aktivira natanko takrat, ko ta ravno zaključi izvajanje trenutnega ukaznega cikla. CPE lahko takoj začne z opravili, povezanimi s skokom v prekinitveni program, saj mu pred tem ni treba čakati zaključka trenutnega ukaznega cikla, ker je ta že zaključen. Najkrajši odzivni čas torej znaša $36 - (50 \cdot 0{,}2) = 26\,\mu$s. Odzivni časi bodo tako variirali v obsegu od 26 do $36\,\mu$s.

c) V povprečju trenutni ukazni cikel, ki mora biti dokončan, doda zakasnitev $(20 \cdot 0{,}2 + 10 \cdot 0{,}3 + 6 \cdot 0{,}5) \cdot 0{,}2 = 28\,\mu$s.

7.14 Režija pri procesiranju prekinitve

V nekem sistemu traja v povprečju $30\,\mu$s do vstopa v ustrezni prekinitveni program, ko je prekinitev potrjena. Izvajanje prekinitvenega programa traja v povprečju $150\,\mu$s. Vendar pa strežba V/I naprave dejansko traja $135\,\mu$s. Preostalih $15\,\mu$s se porabi za povrnitev vsebine registrov CPE in za druge operacije, povezane z vrnitvijo v prekinjeni program [3].

a) Določite skupno režijo (v odstotkih), povezano s prekinitvijo, ob upoštevanju, da je edini produktivni čas tisti, ki se porabi za strežbo naprave.

b) Meritve v realnem sistemu kažejo, da pride v povprečju na sekundo 750 prekinitev. Izračunajte, kolikšen delež časa CPE se porabi za prekinitve.

Rešitev:

a) Iskana režija znaša $\frac{30+15}{30+150} \cdot 100\ \% = 25\ \%$.

b) CPE na sekundo porabi $750 \cdot 180 \cdot 10^{-6}\,\mathrm{s} = 0{,}135\,\mathrm{s}$ časa za prekinitve, kar znaša 13,5 %.

7.15 Vpliv blokovnih prenosov na prekinitvene odzivne čase

Da bi ocenili poslabšanje odzivnosti CPE na zahteve po prekinitvi zaradi prenosov blokov podatkov, ki jih ni mogoče prekiniti, si zamislite sistem, v katerem imajo naprave DMA propustnost 2 MB/s. CPE tega sistema ima ukaze za procesiranje nizov, katerih izvajanja ni mogoče prekiniti. Ta ukaz porabi 15 urinih ciklov za prenos vsake 16-bitne besede [3].

a) CPE uporablja ta ukaz za polnjenje 128-bajtnega vmesnega pomnilnika V/I naprave. Za koliko bi lahko takšni prenosi podaljšali odzivni čas na prekinitve? CPE uporablja uro frekvence 5 MHz.

b) Bloki podatkov, ki jih prenašajo nekatere naprave DMA tega sistema, so dolgi do 256 bajtov. Kakšen je lahko vpliv prenosa bloka ene same naprave DMA na čase odzivov na prekinitve?

c) Predpostavite, da morajo biti prekinitve potrjene najpozneje po 64 μs. Na katero dolžino bi morali omejiti dolžino podatkovnih blokov, da bi izpolnili to zahtevo? Izračunajte ju za oba tipa prenosov, obravnavana v točkah **a)** in **b)**. Ignorirajte čas, ki je potreben za dokončanje trenutnega ukaza.

Rešitev:

a) Časi odziva na prekinitve bi se lahko podaljšali tja do $(128/2) \cdot 15 \cdot 0{,}2 = 192\,\mu$s – to je čas, ki je potreben za napolnitev 128 bajtov vmesnega pomnilnika V/I naprave.

b) Pri hitrosti 2 MB/s traja prenos podatkovnega bloka dolžine 256 bajtov $\frac{256\ \mathrm{B}}{2 \cdot 10^6\ \mathrm{B/s}} = 128\,\mu$s. Odziv na prekinitev bi se zaradi takšnih prenosov DMA lahko zakasnil za do 128 μs.

c) Če se podatki prenašajo z uporabo ukazov za procesiranje nizov, je meja dolžine podatkovnih blokov $\lfloor 128 \cdot \frac{64\,\mu\mathrm{s}}{192\,\mu\mathrm{s}} \rfloor = 42$ B. Če uporabljamo DMA, je meja $128 \cdot \frac{64\,\mu\mathrm{s}}{128\,\mu\mathrm{s}} = 64$ B.

7.16 Vrivanje čakalnih stanj v cikel DMA

Ena izmed ključnih značilnosti krmilnika DMA (DMAC-ja) je hitrost, s katero lahko prenaša podatke v pomnilnik ali iz pomnilnika. Vendar pa je dejanska hitrost DMA-ja odvisna tudi od hitrosti pomnilnika. Pogosto je treba upočasnjevati DMAC, tako da mu pomnilnik lahko sledi. DMAC lahko upočasnujemo bodisi z zniževanjem frekvence ure ali z vstavljanjem čakalnih stanj. Za primer vzemimo čip 8237A DMAC. Tako kot pri procesorjih 8088/8086 trajajo njegovi bralni in pisalni cikli štiri periode urinega signala. Čip 8237A vzorči svoj vhod READY v periodi T3 in vrine eno čakalno stanje, če je v nizkem stanju. Od takrat naprej vriva čakalna stanja vse dotlej, dokler READY ne gre na visoko stanje [3].

a) Predpostavite, da je pomnilnik hitrejši od DMAC-ja. Kolikšna je najvišja mogoča hitrost prenosov DMA, če je frekvenca ure DMAC-ja 4 MHz?

b) Če je frekvenca ure DMAC-ja 4 MHz, DMAC zagotavlja, da je širina impulza za branje 220 ns. Pomnilnik pa zahteva, da je ta parameter vsaj 350 ns. Koliko čakalnih stanj moramo vriniti v vsak bralni cikel DMA? Kakšna je posledična hitrost prenosa DMA?

c) Informacija v priročniku za DMAC zagotavlja, da je širina prožilnega signala za branje enaka T − 30 ns, kjer je T perioda ure DMAC-ja. Ali lahko izpolnimo zahtevo po širini prožilnega signala za branje iz pomnilnika brez vrivanja čakalnih stanj? Do katere vrednosti naj bi znižali frekvenco ure DMAC-ja, da bi zadostili zahtevi?

Rešitev:

a) Pri frekvenci 4 MHz znaša perioda ure 250 ns. Ker vsak cikel DMA traja štiri periode urinega signala, trajajo cikli DMA $4 \cdot 250\,\text{ns} = 1.000\ \text{ns} = 1\,\mu\text{s}$. Tako zmore DMAC izvesti 1 milijon ciklov DMA na sekundo.

b) Pomnilnik je prepočasen za $350 - 220 = 130$ ns. Tako bo eno čakalno stanje, ki podaljša cikel DMA za 250 ns, več kot dovolj. Cikel DMA traja zdaj $5 \cdot 250\,\text{ns} = 1.250\,\text{ns} = 1{,}25\,\mu\text{s}$, kar pomeni 0,8 milijona ciklov DMA na sekundo.

c) Zahtevo lahko izpolnimo, če podaljšamo periodo ure, tako da bo izpolnjena neenačba $\text{T} - 30\,\text{ns} \geq 350\,\text{ns}$. To pomeni, da mora biti T vsaj 380 ns, kar ustreza frekvenci ure $1/(380 \cdot 10^{-9}\ \text{s}) \approx 2{,}6\,\text{MHz}$.

7.17 Prenos nadzora nad vodilom med CPE in DMAC

Ko je naprava pripravljena na prenos DMA, generira zahtevo DMA k DMAC-ju. Nato DMAC aktivira linijo BR („bus request“) CPE, da zahteva nadzor nad vodilom. Pretečeni čas, dokler se CPE ne odzove s signalom BG („bus grant“), ni odvisen samo od določene CPE, ampak tudi od tega, kaj CPE dela v trenutku, ko DMAC postavi svojo zahtevo. Na primer, če je CPE zaposlena s ciklom na vodilu, bo počakala na njegovo zaključitev in šele nato odobrila vodilo. Tukaj predpostavimo, da se vodilo odobri znotraj dveh urinih ciklov. Podobno tudi tedaj, ko DMAC deaktivira signal BR, CPE signala BG ne deaktivira takoj. To stori po preteku dveh urinih ciklov. Predpostavite, da cikel DMA traja štiri urine cikle in da tako CPE kot DMAC uporabljata uro frekvence 5 MHz [3].

a) Koliko časa traja prenos 64 bajtov, če deluje DMAC v načinu kraje ciklov? Upoštevajte tudi čas za prevzem in sprostitev vodila.

b) Ponovite izračun za delovanje v eksplozijskem načinu.

c) Kolikšen delež izračunanih vrednosti predstavlja režijo.

Rešitev:

a) Pri frekvenci 5 MHz je perioda ure 200 ns. Za prenos enega bajta potrebujemo $2 + 4 + 2 = 8$ urinih ciklov, kar pomeni $8 \cdot 0{,}2\,\mu\text{s} = 1{,}6\,\mu\text{s}$. Prenos 64 bajtov bo trajal $64 \cdot 1{,}6\,\mu\text{s} = 102{,}6\,\mu\text{s}$, če predpostavimo, da naprava DMA prenaša podatke po en bajt naenkrat.

b) V eksplozijskem načinu traja prenos $(2 + 64 \cdot 4 + 2) \cdot 0{,}2\,\mu\text{s} = 52\,\mu\text{s}$.

c) V načinu kraje ciklov je režija $(4/8) \cdot 100\ \% = 50\ \%$. V eksplozijskem načinu je režija samo $(4/256) \cdot 100\ \% = 1{,}56\ \%$.

7.18 Programski V/I proti prekinitveno vodenemu V/I

Neka počasna V/I naprava preskrbi kvečjemu en bajt informacije vsakih 100 ms. CPE preverja status naprave redno približno na vsakih 50 ms (programski V/I). Nekatere analize kažejo, da za vsakokratno preverjanje statusa naprave CPE porabi okrog 20 μs svojega časa. Poleg tega je bilo ugotovljeno, da naprava v povprečju preskrbi samo en bajt na sekundo. Izračunajte, za koliko odstotkov se skrajša poraba časa CPE za komunikacijo s to napravo, če slednjo opremimo z zmožnostjo prekinitev. Predpostavite, da prekinitveni program in pridružena režija znašata 40 μs [3].

Rešitev:
Izračunali bomo ustrezna časa CPE, porabljena v intervalu ene sekunde za oba primera. V načinu s programskim V/I opravi CPE v eni sekundi (1.000 ms/50 ms) = 20 preverjanj statusa naprave. Za to porabi $20 \cdot 20\,\mu\text{s} = 400\,\mu\text{s}$. Če je V/I naprava zmožna delati s prekinitvami, porabi CPE samo 40 μs. To predstavlja zmanjšanje porabe časa CPE za komunikacijo z V/I napravo za $\frac{400-40}{400} \cdot 100\ \% = 90\ \%$.

7.19 Prekinitveno voden V/I v primerjavi z DMA

Neka V/I naprava je zmožna sprejemati en bajt podatka vsakih 100 μs. Podatki se prenašajo na napravo v blokih po 256 bajtov. Najprej razmislite o zasnovi sistema tako, da bo naprava povzročila prekinitev, kadarkoli je pripravljena sprejeti naslednji bajt podatkov. Ocenjeno je, da strežba vsake prekinitve traja skupno okrog 40 μs [3].

a) Kakšen je koeficient izboljšanja, če uporabljamo DMA v načinu kraje ciklov in prekinemo CPE samo po prenosu celotnega bloka? Predpostavite, da se porabi 2 μs časa za pridobitev vodila, prenos bajta in povrnitev nadzora nad vodilom nazaj k CPE. Predpostavite tudi, da CPE ne more delati nič koristnega, dokler je vodilo zasedeno.

b) Predpostavite, da so takšni 256 bajtov dolgi bloki prenašani k napravi s povprečno hitrostjo enega bloka na vsaki 2 sekundi. Kakšen delež časa CPE se porabi za strežbo naprave, če uporabljamo prekinitveno voden V/I?

c) Predpostavite, da se bloki prenašajo s hitrostjo 20 blokov na sekundo. Kolikšen delež časa CPE bomo prihranili, če bomo uporabili DMA?

Rešitev:

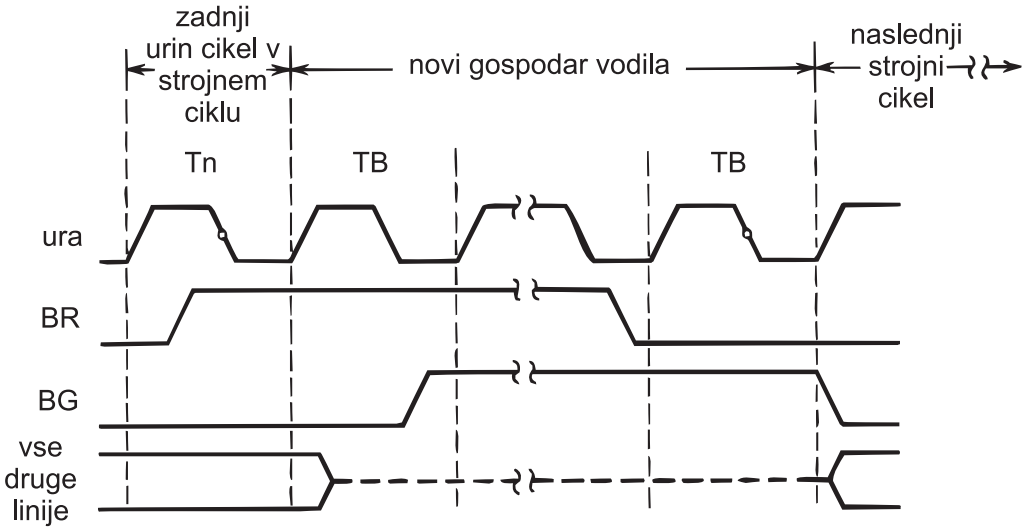
a) Prekinitveno voden V/I zahteva za prenos bloka $256 \cdot 0{,}04\,\text{ms} = 10{,}24\,\text{ms}$ časa CPE. Z DMA-jem traja prenos samo $256 \cdot 0{,}002\,\text{ms} + 0{,}04\,\text{ms} = 0{,}552\,\text{ms}$. Koeficient izboljšanja (skrajšanje časa prenosa bloka) znaša $10{,}24/0{,}552 = 18{,}55$.

b) Prenos enega bloka traja $256 \cdot 40\,\mu\text{s} = 10{,}24\,\text{ms}$. Ker sta zaporedna bloka med seboj oddaljena za 2 s, znaša delež časa CPE, ki se porabi za strežbo naprave, $\frac{10{,}24\,\text{ms}}{2.000\,\text{ms}} \cdot 100\ \% = 0{,}512\ \%$.

c) Pri hitrosti prenosa 20 blokov na sekundo znaša čas med dvema zaporednima blokoma $1\,\text{s}/20 = 50\,\text{ms}$. Če uporabljamo prekinitveno voden V/I, znaša delež časa CPE, ki se porabi za strežbo naprave, $\frac{10{,}24\,\text{ms}}{50\,\text{ms}} \cdot 100\ \% = 20{,}48\ \%$. Z uporabo DMA-ja pa znaša delež časa CPE za strežbo naprave zgolj $\frac{10{,}552\,\text{ms}}{50\,\text{ms}} \cdot 100\ \% = 1{,}1\ \%$ kar pomeni, da smo prihranili kar 20,48 % - 1,1 % = 19,38 % časa CPE.

7.20 Čas prenosa nadzora nad vodilom

Časovni potek cikla za dodelitev vodila nekega mikroprocesorja je identičen s tem na spodnji sliki. Novi gospodar prevzame vodilo z namenom, da prenese v pomnilnik blok z 256 bajti. Prenos vsakega bajta traja dve stanji TB. Koliko je dolg časovni interval z zaporednimi stanji TB, če je frekvenca ure 8 MHz [3]?



Rešitev:
Novi gospodar vodila bo porabil $2 \cdot 256 = 512$ stanj TB za prenos podatkov v pomnilnik. Pred prenosom traja eno stanje TB, da novi gospodar pridobi nadzor nad vodilom, po prenosu pa še eno stanje TB, da preda nadzor nad vodilom prejšnjemu gospodarju. Iz tega sledi, da je skupno število stanj TB enako $1 + 512 + 1 = 514$. Časovni interval z zaporednimi stanji TB traja $514 \cdot 125\,\text{ns} = 64.250\,\text{ns} = 64{,}25\,\mu\text{s}$.

7.21 Prioriteta dostopov do pomnilnika med CPE in DMA

V skoraj vseh sistemih, ki vsebujejo module DMA, se dostopom DMA-ja do pomnilnika daje višja prioriteta kot dostopom CPE do pomnilnika. Zakaj [1]?

Rešitev:
Če se procesor nekoliko zadrži pri poskusu branja iz pomnilnika ali pisanja v pomnilnik, običajno ne pride do kakšne škode, ampak le do majhne izgube časa. Po drugi strani pa se lahko prenos DMA bodisi k V/I napravi ali od V/I naprave izvaja v obliki toka podatkov (npr. trdi disk ali magnetni trak), ki ga ni mogoče ustaviti. Če je torej modul DMA zadržan (zavrnjen nadaljnji dostop do glavnega pomnilnika), bodo podatki izgubljeni.

7.22 Upočasnitev CPE zaradi načina DMA s krajo ciklov

Nek modul DMA prenaša znake z naprave, na kateri se podatki generirajo s hitrostjo 9.600 bit/s, v pomnilnik z uporabo kraje ciklov. Procesor včituje ukaze s hitrostjo 1 milijon ukazov na sekundo (1 MIPS). Za kolikšen delež časa bo procesor upočasnjen zaradi aktivnosti DMA-ja [1]?

Rešitev:
Zanemarimo operacije branja in pisanja podatkov in predpostavimo, da procesor samo včituje ukaze. Potem procesor potrebuje dostop do glavnega pomnilnika enkrat na vsako mikrosekundo. Modul DMA prenaša znake s hitrostjo $9.600/8 = 1.200$ znakov na sekundo, kar pomeni enega na vsako $1/1.200$ sekunde ali na vsakih 833 μs. DMA torej ukrade procesorju vsaki 833. cikel. To upočasni procesor za približno $(1/833) \cdot 100\ \% = 0{,}12\ \%$.

7.23 Zakasnitev prenosa DMA zaradi zasedenosti vodila

Cikel na vodilu neke CPE je sestavljen iz šestih urinih ciklov. Zaradi dolžine časa dostopa do pomnilnika mora CPE vrivati po dve čakalni stanji v vsak cikel na vodilu. Ko naprava DMA prek svojega vmesnika aktivira linijo BR, traja dva urina cikla, da si pridobi nadzor nad vodilom, če na vodilu trenutno ni nobene aktivnosti (CPE prek vodila ne izvaja nobenega prenosa) [3].

a) Koliko časa mora taka naprava čakati, če CPE poganjamo z uro frekvence 8 MHz?

b) Ali bi uporabili DMA v eksplozijskem načinu za napravo, ki prenaša podatke s hitrostjo 0,75 MB/s?

Rešitev:

a) Predpostavimo najprej, da se trenutno na vodilu ne izvaja noben cikel. Urin cikel traja 125 ns. V/I naprava bo torej čakala $2 \cdot 125\,\text{ns} = 250\,\text{ns}$ za prevzem nadzora na vodilu. Če pa se na vodilu trenutno izvaja cikel, bi V/I naprava morala čakati največ še dodatnih $(6+2) \cdot 125\,\text{ns} = 1.000\,\text{ns} = 1\,\mu\text{s}$. V/I naprava bo na začetek prenosa DMA morala čakati za časovni interval med 250 ns in 1,25 μs.

b) Če naprava prenaša podatke s hitrostjo 0,75 MB/s, sta zaporedna bajta oddaljena 1,33 μs. Ta čas je prekratek, da bi CPE lahko pred prihodom naslednjega bajta napravila sploh kakšen ukazni cikel, zato način kraje ciklov ne pride v poštev. Za V/I napravo s podano hitrostjo bi uporabili DMA v eksplozijskem načinu.

7.24 Primerjava eksplozijskega načina in načina kraje ciklov

Imamo sistem, kjer cikli na vodilu trajajo 500 ns. Prenos nadzora nad vodilom v katerikoli smeri (od procesorja k V/I napravi ali obratno) traja 250 ns. Ena izmed V/I naprav sistema ima hitrost prenosa 50 kB/s in uporablja DMA. Podatki se prenašajo po bajtih (en bajt naenkrat) [1].

a) Predpostavite, da uporabljamo DMA v eksplozijskem načinu. To pomeni, da vmesnik DMA-ja pridobi nadzor nad vodilom pred začetkom prenosa bloka in zadrži nadzor nad vodilom, dokler ne prenese celotnega bloka. Kako dolgo bi naprava vezala vodilo pri prenosu bloka s 128 bajti?

b) Ponovite izračun za način DMA-ja s krajo ciklov.

Rešitev:

a) Prenos 128 bajtov podatkov traja $\frac{128\ \mathrm{B}}{50\ \mathrm{kB/s}} = 2{,}56\,\mathrm{ms}$. Temu moramo prišteti še čas za prenos nadzora nad vodilom na začetku in po koncu prenosa, kar znaša $250\,\mathrm{ns} + 250\,\mathrm{ns} = 500\,\mathrm{ns}$. Ta dodatni čas je proti času samega prenosa podatkov zanemarljiv, zato lahko zaključimo, da bo vodilo vezano za 2,56 ms.

b) Čas, ki je potreben za prenos enega bajta v načinu kraje ciklov, je $250\,\mathrm{ns} + 500\,\mathrm{ns} + 250\,\mathrm{ns} = 1.000\,\mathrm{ns} = 1\,\mu\mathrm{s}$. Ker se med zaporednimi prenosi bajtov vodilo sprosti, je zasedeno samo $128 \cdot 1\,\mu\mathrm{s} = 128\,\mu\mathrm{s}$. Izbira DMA-ja v načinu kraje ciklov nam torej zmanjša zasedenost vodila v primerjavi z delovanjem DMA-ja v eksplozijskem načinu za faktor $\frac{2{,}56\,\mathrm{ms}}{128\,\mu\mathrm{s}} = 20$.

7.25 Krmilnik DMA Intel 8237A (1/2)

Pregled časovnih diagramov za krmilnik DMA Intel 8237A kaže, da traja cikel DMA-ja na vodilu, v katerem prenese en bajt podatkov med pomnilnikom in V/I napravo, tri periode ure [3].

a) Predpostavite, da krmilnik 8237A uporablja uro frekvence 5 MHz. Koliko traja prenos enega bajta?

b) Kakšna bi bila največja možna hitrost prenosa?

c) Predpostavite, da pomnilnik ni dovolj hiter in moramo vriniti v vsak cikel DMA dve čakalni stanji. Kakšna bo dejanska hitrost prenosa?

Rešitev:

a) Pri frekvenci ure 5 MHz traja urin cikel 0,2 μs. Prenos enega bajta traja torej $3 \times 0{,}2\,\mu\mathrm{s} = 0{,}6\,\mu\mathrm{s}$.

b) Najvišja hitrost prenosa, ki jo lahko dosežemo, je $\frac{1\,\mathrm{B}}{0{,}6 \cdot 10^{-6}\,\mathrm{s}} = 1{,}67\,\mathrm{MB/s}$.

c) Če je čas pomnilniškega cikla daljši od 600 ns, moramo vriniti čakalna stanja z manipulacijo linije READY na 8237A. Upočasniti moramo cikle DMA. Vrivanje dveh čakalnih stanj v vsak cikel DMA bo podaljšalo cikel DMA na $(3+2)\cdot 0{,}2\,\mu\text{s} = 1\,\mu\text{s}$. Rezultat tega je, da bo hitrost prenosa padla na 1 MB/s.

7.26 Krmilnik DMA Intel 8237A (2/2)

Predpostavite, da v sistemu iz primera **7.25** pomnilniški cikel traja 750 ns. Do katere vrednosti lahko zmanjšamo frekvenco ure vodila brez vpliva na doseženo hitrost prenosa [3]?

Rešitev:
Cikel DMA lahko traja do 750 ns brez potrebe po vrivanju čakalnih stanj. To ustreza periodi ure $750/3 = 250$ ns, kar pomeni frekvenco ure 4 MHz. Ta pristop bi odpravil potrebo po vezju za vrivanje čakalnih stanj in tudi zmanjšal porabo moči.

7.27 Izbira načina delovanja krmilnika DMA

Nek krmilnik DMA streže štirim sprejemnim telekomunikacijskim povezavam (ena povezava na vsak kanal DMA) s hitrostjo 64 kb/s (kilobitov na sekundo) na povezavo [1].

a) Ali bi uporabljali krmilnik DMA v eksplozijskem načinu ali v načinu kraje ciklov?

b) Katero shemo prioritete bi izkoristili za strežbo kanalov DMA?

c) Predpostavite delovanje DMA s krajo ciklov. Prenos enega bajta traja $2\,\mu$s. Koliko časa ostane CPE vodilo na voljo za druga opravila?

Rešitev:

a) Telekomunikacijska povezava lahko prenaša podatke nepretrgoma, zato eksplozijski način ne pride v poštev, saj krmilnik DMA sploh ne bi dal procesorju nobene priložnosti, da bi prišel do vodila. Potreben je torej DMA s krajo ciklov.

b) Ker imajo vse linije enako hitrost prenašanja podatkov, bi vsem dodelili enako prioriteto. To pomeni izbiro prioritetne sheme z rotiranjem.

c) Hitrost 64 kb/s ustreza 8 kB/s, to je en bajt vsakih $125\,\mu$s prek vsake linije. V časovnem intervalu $125\,\mu$s naj bi krmilnik DMA izvedel prenos vsaj štirih bajtov, kar pomeni, da ima CPE za druga opravila vodilo na voljo kar $125\,\mu\text{s} - 4\cdot 2\,\mu\text{s} = 117\,\mu$s.

7.28 Selektorski in multipleksorski kanal

Nek 32-bitni računalnik ima dva selektorska kanala in en multipleksorski kanal. Vsak selektorski kanal podpira dva magnetna diska in dve magnetni tračni enoti. Na multipleksorski kanal sta povezana dva linijska tiskalnika, dva čitalnika kartic in 10 videoterminalov. Predpostavite naslednje prenosne hitrosti:

diskovni pogon	800 kB/s,
magnetna tračna enota	200 kB/s,
linijski tiskalnik	6,6 kB/s,
čitalnik kartic	1,2 kB/s,
videoterminal	1 kB/s.

Ocenite najvišjo skupno V/I hitrost v tem sistemu [1].

Rešitev:
Na selektorskem kanalu je lahko sočasno postrežena samo ena naprava, zato je najvišja skupna V/I hitrost $800 + 800 + 2 \cdot 6{,}6 + 2 \cdot 1{,}2 + 10 \cdot 1 = 1.625{,}6$ kB/s.

7.29 Primerjava programskega V/I in DMA

Računalnik je sestavljen iz procesorja in V/I naprave D, povezane na glavni pomnilnik M prek skupnega vodila s podatkovnim vodilom širine ene besede. Procesor lahko izvaja največ 10^6 ukazov na sekundo. Povprečni ukaz zahteva pet strojnih ciklov, od katerih trije uporabljajo pomnilniško vodilo. Operacije branja iz pomnilnika in pisanja v pomnilnik zahtevajo en strojni cikel. Predpostavite, da procesor neprekinjeno izvaja procese v ozadju, ki zahtevajo 95 % njegovega izvajalnega časa, vendar ne vsebujejo nobenega V/I ukaza. Denimo, da je trajanje enega procesorskega cikla enako enemu ciklu na vodilu. Zdaj predpostavite, da uporabljamo V/I napravo za prenos zelo velikih blokov podatkov med M in D [1].

a) Ocenite najvišjo V/I hitrost prenosa podatkov v besedah na sekundo, ki je mogoča skozi D, če se uporablja programski V/I in vsak enobesedni V/I prenos zahteva izvedbo dveh procesorskih ukazov.

b) Ocenite to isto hitrost, če se uporablja DMA.

Rešitev:

a) Procesor lahko samo 5 % svojega časa posveti za V/I. Torej je najvišja hitrost izvajanja V/I ukazov $10^6 \cdot 0{,}05 = 50.000$ ukazov na sekundo. V/I prenosna hitrost je torej 25.000 besed na sekundo.

b) Število strojnih ciklov v eni sekundi, ki so na razpolago za DMA, je enako $10^6 \cdot (0{,}05 \cdot 5 + 0{,}95 \cdot 2) = 2{,}15 \cdot 10^6$. Če vzamemo, da lahko modul DMA uporablja vse te cikle, ter zanemarimo morebitni nastavitveni čas in čas za preverjanje statusa, potem najvišja V/I prenosna hitrost znaša $2{,}15 \cdot 10^6$ besed na sekundo.

7.30 Hitrost prenosa podatkov pri serijski komunikaciji

Podatkovni vir proizvaja 7-bitne znake IRA (International Reference Alphabet), ki jim je pripet paritetni bit. Izpeljite izraz za najvišjo dejansko hitrost prenosa podatkov ER (Effective data Rate) prek linije s pasovno širino R bitov na sekundo (bps) za naslednje primere [1]:

a) asinhroni prenos s končnim bitom, ki traja poldrugo dolžino bitne celice,

b) bitno usmerjeni sinhroni prenos z okvirjem, sestavljenim iz 48 krmilnih bitov in 128 informacijskih bitov,

c) enako kot v **b)**, le da je informacijsko polje dolgo 1.024 bitov,

d) znakovno usmerjeni sinhroni prenos z devetimi krmilnimi znaki na okvir in 16 informacijskimi znaki,

e) enako kot v **d)**, le da je v okvirju 128 informacijskih znakov.

Rešitev:
Za vsak primer izračunamo delež g prenesenih bitov, ki so podatkovni biti. Potem je najvišja dejanska hitrost prenosa podatkov enaka ER $= gR$.

a) Imamo 7 podatkovnih bitov, 1 začetni bit, končni bit dolžine poldruge bitne celice in 1 paritetni bit.

$$\text{ER} = gR = \frac{7}{1+7+1+1{,}5}\cdot R = 0{,}67\cdot R$$

b) Vsak okvir vsebuje $48 + 128 = 176$ bitov. Število znakov je $128/8 = 16$, število podatkovnih bitov pa $16 \cdot 7 = 112$.

$$\text{ER} = \frac{112}{176}\cdot R = 0{,}64\cdot R$$

c) Vsak okvir vsebuje $48 + 1.024 = 1.072$ bitov. Število znakov je $1.024/8 = 128$, število podatkovnih bitov pa $128 \cdot 7 = 896$.

$$\text{ER} = \frac{896}{1.072}\cdot R = 0{,}84\cdot R$$

d) Z devetimi krmilnimi znaki in 16 informacijskimi znaki vsebuje vsak okvir $(9 + 16) \cdot 8 = 200$ bitov. Število podatkovnih bitov je $16 \cdot 7 = 112$.

$$\text{ER} = \frac{112}{200}\cdot R = 0{,}56\cdot R$$

e) Z devetimi krmilnimi znaki in 128 informacijskimi znaki vsebuje vsak okvir $(9 + 128) \cdot 8 = 1.096$ bitov. Število podatkovnih bitov je $128 \cdot 7 = 896$.

$$\text{ER} = \frac{896}{1.096}\cdot R = 0{,}82\cdot R$$

7.31 Hitrost naraščanja/padanja signala po EIA RS-232C

Standard Združenja elektronske industrije EIA RS-232C med drugim omejuje hitrost naraščanja/padanja električnega signala (slew rate) na izhodu gonilnika, ki ne sme biti večja od $30\,\text{V}/\mu\text{s}$. Ideja je upočasniti prehode signala in s tem zmanjšati presluh k drugim signalnim linijam. Če kabel gonilniku ne predstavlja dovolj kapacitivnosti, moramo dodati zunanji kondenzator C_1 [3].

a) Denimo, da se signal na izhodu gonilnika spreminja med $-11{,}75\,\mathrm{V}$ in $11{,}75\,\mathrm{V}$. Prehodi (v vsaki smeri) so skoraj linearni in trajajo $0{,}5\,\mu\mathrm{s}$. Ali prehodi signala izpolnjujejo omejitev RS-232C za hitrost naraščanja/padanja električnega signala?

b) Da bi izpolnili zahteve standarda RS-232C za hitrost naraščanja/padanja električnega signala, se odločimo dodati zunanji kondenzator C_1. Kakšna naj bo njegova približna vrednost, če gonilnik daje konstanten tok okrog $10\,\mathrm{mA}$?

c) Koliko časa traja prehod od $-11{,}75\,\mathrm{V}$ do $11{,}75\,\mathrm{V}$ po priključitvi takšnega kondenzatorja na izhod vsakega gonilnika?

d) V splošnem se časi vzpona merijo med točkama 10 % in 90 % prehoda. Izračunajte čas vzpona brez in čas vzpona z dodanim kondenzatorjem C_1 na izhodu gonilnika.

Rešitev:

a) Med prehodom se izhodna napetost spremeni za skupno $2 \cdot 11{,}75\,\mathrm{V} = 23{,}5\,\mathrm{V}$. Hitrost naraščanja/padanja električnega signala je $\frac{23{,}5\,\mathrm{V}}{0{,}5\,\mu\mathrm{s}} = 47\,\mathrm{V}/\mu\mathrm{s}$. Prehodi signala torej ne izpolnjujejo zahteve za RS-232C.

b) Zahtevano kapacitivnost lahko določimo iz enačbe $q = CV$, kjer je q naboj in V napetost na kondenzatorju C. Z odvajanjem obeh strani enačbe po času dobimo $\frac{dq}{dt} = C\frac{dV}{dt}$ ali $i = C\frac{dV}{dt}$, kjer je i tok skozi kondenzator. Za C dobimo

$$C = \frac{i}{\frac{dV}{dt}} = \frac{10 \cdot 10^{-3}\,\mathrm{A}}{30\,\frac{\mathrm{V}}{\mu\mathrm{s}}} \approx 330\,\mathrm{pF}$$

Če želimo izpolniti zahteve, mora gonilnik na svojem izhodu čutiti skupno kapacitivnost vsaj $330\,\mathrm{pF}$. Dejansko kapacitivnost, ki jo čuti gonilnik, izračunamo po isti enačbi, če vstavimo za $\frac{dV}{dt}$ vrednost $47\,\mathrm{V}/\mu\mathrm{s}$. Rezultat je okrog $210\,\mathrm{pF}$. K tej kapacitivnosti v glavnem prispeva kabel. Da bomo zadovoljili zahteve za RS-232C, moramo dodati zunanji kondenzator C_1 s kapacitivnostjo vsaj $330\,\mathrm{pF} - 210\,\mathrm{pF} = 120\,\mathrm{pF}$.

c) Nalogo si poenostavimo, če predpostavimo, da so prehodi signala linearni. Iz primerjave dolžin katet dveh pravokotnih trikotnikov lahko zapišemo

$$\frac{2 \cdot 11{,}75\,\mathrm{V}}{t} = \frac{30\,\mathrm{V}}{1\,\mu\mathrm{s}}.$$

Iz te enačbe lahko izpeljemo, da prehod signala (t) od $-11{,}75\,\mathrm{V}$ do $11{,}75\,\mathrm{V}$ traja $0{,}783\,\mu\mathrm{s}$.

d) Točki 10 % in 90 % napetosti znašata $2{,}35\,\mathrm{V}$ in $21{,}15\,\mathrm{V}$. Sprememba napetosti znaša torej $21{,}15 - 2{,}35 = 18{,}8\,\mathrm{V}$. Čas vzpona pred priključitvijo kondenzatorja C_1 je potemtakem $t_{r1} = \frac{18{,}8\,\mathrm{V}}{47\,\mathrm{V}/\mu\mathrm{s}} = 0{,}4\,\mu\mathrm{s}$, čas vzpona po priključitvi kondenzatorja pa je $t_{r2} = \frac{18{,}8\,\mathrm{V}}{30\,\mathrm{V}/\mu\mathrm{s}} = 0{,}63\,\mu\mathrm{s}$.

7.32 Asinhrona serijska komunikacija

Predpostavite, da je frekvenca oddajne in sprejemne ure nekega serijskega komunikacijskega krmilnika (SCC-ja) 38,4 kHz. Napravo smo inicializirali za Baudno hitrost s faktorjem ×16, kar pomeni, da ima ura na vhodu 16-krat višjo frekvenco kot dejanska Baudova hitrost. SCC signal ure na vhodu deli z Baudovim faktorjem 16 in tako dobi 16 notranjih period ure na vsak bit, ki omogočajo natančno vzorčenje. Kdaj najpozneje mora CPE prebrati sprejet znak, da se ne bo izgubil? Serijski komunikacijski krmilnik smo inicializirali tako, da znak vsebuje osem podatkovnih bitov, pariteto in dva končna bita [3].

Rešitev:
Ker je faktor Baudne hitrosti ×16, je dejanska hitrost prenosa šestnajstkrat nižja od frekvence ure in tako znaša $\frac{38.400}{16} = 2.400\,\mathrm{b/s}$. Vsak znak je sestavljen skupno iz 12 bitov. Najvišja hitrost prenosa znakov je torej $\frac{2.400}{12} = 200$ znakov/s. To pomeni, da so zaporedni znaki lahko do $\frac{1\,\mathrm{s}}{200} = 0{,}005\,\mathrm{s}$, to je 5 ms vsaksebi. Po obvestilu o prejemu znaka ga mora CPE prebrati v naslednjih 5 ms, sicer tvegamo, da bo izgubljen.

7.33 Prekinitveno voden V/I za serijsko komunikacijo

Nek serijski komunikacijski krmilnik dela s hitrostjo 2.400 b/s. Vsak znak je sestavljen iz 12 bitov [3].

a) Kolikšna je najvišja pogostnost, s katero bi krmilnik prekinjal CPE?

b) Določite povprečno pogostnost prekinjanja CPE, če je povprečna izkoriščenost komunikacijskega kanala samo 0,2.

Rešitev:

a) Iz dane bitne hitrosti in formata izračunamo najvišjo hitrost prenosa, ki znaša $\frac{2.400}{12} = 200$ znakov/s – to je, en znak vsakih 5 ms. Pri polni dupleksni komunikaciji to velja za vsako smer.

b) Pogostnost prekinjanja znaša do 400 prekinitev na sekundo ali eno prekinitev na 2,5 ms. Pri danih pogojih se ta pogostnost zmanjša na $400 \cdot 0{,}2 = 80$ prekinitev na sekundo.

7.34 Časovne zanke

Včasih moramo generirati časovne zakasnitve specificiranih dolžin za učasovanje določenih akcij v izvajanju programa. Seveda lahko takšne časovne zakasnitve generiramo s programirljivimi časovniki. Drugačen pristop je uporaba programskih zank, katerih čas izvajanja lahko izračunamo iz časov izvajanja posameznih ukazov. Določena časovna zakasnitev se nato ustvari z ustreznim številom izvedb časovne zanke [3].

a) Iz uporabniškega priročnika CPE ugotovite čas izvajanja (v številu urinih ciklov) vsakega ukaza v zanki. Nato s seštevanjem izračunate, da izvajanje zanke traja 80 urinih ciklov. Kolikokrat bi morali izvesti zanko, če morate ustvariti časovno zakasnitev 1 ms? Predpostavite, da CPE deluje s frekvenco ure 8 MHz.

b) Kako bi generirali 250 μs širok impulz na eni izmed linij izhodnih vrat?

c) Predpostavite, da nadgradite sistem z različico CPE, ki deluje na frekvenci 10 MHz. Kakšen bo novi čas izvajanja časovne zanke? Kakšne možnosti prilagoditev časovne zanke imate na voljo, če želite preprečiti časovne probleme v točki **b)**?

Rešitev:

a) Pri frekvenci 8 MHz znaša perioda ure 125 ns = 0,125 μs. Izvajanje zanke traja tako $80 \cdot 0{,}125\,\mu\text{s} = 10\,\mu\text{s}$. Za časovno zakasnitev 1 ms se mora zanka izvesti $1.000/10 = 100$-krat.

b) En bit izhodnih vrat bi postavili na vrednost 1, zanko bi izvedli $250/10 = 25$-krat in nato resetirali ta bit nazaj na vrednost 0.

c) Čas izvajanja zanke je zdaj $80 \cdot 0{,}1\,\mu\text{s} = 8\,\mu\text{s}$. Da bi preprečili časovne probleme, moramo bodisi dodati določeno število ukazov NOP (no operation) za podaljšanje časa izvajanja zanke nazaj na 10 μs ali povečati število izvajanj zanke na 250/8, to je na 31.

7.35 Programirljivi časovniki

Po možnosti časovne zakasnitve ustvarimo s strojno in ne s programsko opremo (glejte nalogo **7.34**). Ponavadi to izvedemo s programirljivimi časovniki. Poleg tega, da odpravijo neposredno sodelovanje CPE, časovniki ponujajo tudi mnogo drugih koristnih funkcij. Zamislimo si sistem, ki uporablja programirljivi intervalni časovnik (PIT) Intel 8254. Časovnik uporablja uro frekvence 2 MHz. Vsebuje tri 16-bitne nastavljive sinhrone števnike, ki štejejo navzdol bodisi binarno ali v kodu BCD (dvojiško kodirano desetiško število) [3].

a) Kakšno začetno vsebino morate naložiti v prvi števnik, če ga želite uporabiti za generiranje prekinitev na vsako 1 ms? Kolikšen je najdaljši časovni interval, ki ga lahko generira ta števnik, če šteje v kodu BCD?

b) Predpostavite, da je vaš sistem povezan z omrežjem. Poslati morate neko sporočilo k drugemu sistemu in želite izmeriti, koliko časa preteče, da prejmete odgovor nazaj. Opišite, kako bi izvedli takšno meritev.

c) Kolikšen je najdaljši časovni interval, ki ga lahko izmerite s časovnikom Intel 8254?

Rešitev:

a) Pri frekvenci 2 MHz znaša perioda ure za proženje časovnika 0,5 μs. Da bi se časovnik pri takšnem taktu ure po inicializaciji iztekel (vrebina v števniku doseže vrednost 0) po preteku časovnega intervala 1 ms, ga moramo inicializirati z začetno vrednostjo $1.000/0{,}5 = 2.000$. Najdaljši časovni interval, ki ga s tem časovnikom lahko dosežemo v načinu BCD, ustreza največjemu BCD številu, ki ga lahko vpišemo v 16-bitni BCD števnik. To število je 9.999, ki ustreza časovnemu intervalu $9.999 \cdot 0{,}5\,\mu\text{s} = 4.999{,}5\,\mu\text{s} \approx 5\,\text{ms}$.

b) V števnik vpišemo največjo vrednost takoj, ko sporočilo odpošljemo. Ko prejmemo odgovor, preberemo vsebino števnika, jo odštejemo od največje vrednosti in razliko pomnožimo s periodo urinega signala.

c) Če povežemo vse tri časovnike, lahko merimo časovne intervale do $2^{48}/f$, kar pri frekvenci 2 MHz znaša več kot 4 leta in 5 mesecev.

7.36 Sinhrona serijska komunikacija*

Nek sistem je opremljen z dvoje sinhronih serijskih komunikacijskih vrat, delujočih v načinu polnega dupleksa. Prenosna hitrost obeh (dvosmernih) kanalov je 9.600 b/s. Vsak informacijski okvir je dolg 1.064 bitov in nosi 128 bajtov podatkov [3].

a) Serijski komunikacijski krmilnik (SCC) je sprogramiran tako, da prekine CPE vsakokrat, ko je pripravljen sprejeti ali preskrbeti nov bajt podatkov. Kako pogosto bo SCC prekinjal CPE med sprejemanjem okvirja? Kaj pa, če SCC hkrati obravnava en okvir na smer vsakega kanala?

b) Ocenjeno je, da procesiranje prekinitve (za vsak podatkovni bajt) traja skupaj okrog 50 μs in da sestavljen promet (prek obojih vrat) znaša 25 okvirjev na sekundo. Kolikšen delež svojega časa porabi CPE za strežbo SCC-ja?

c) Predpostavite, da uporabljamo DMA in je CPE še vedno prekinjana enkrat na okvir. Podatki se prenašajo v pomnilnik ali iz pomnilnika s štirikanalnim DMAC-jem. V načinu kraje ciklov vsak cikel DMA traja 1 μs (vključno z režijo za prenos nadzora na vodilu). Ocenjeno je tudi, da med takšnimi prenosi obstaja 20-odstotna verjetnost, da bo procesor ostal nedejaven in čakal, da DMAC sprosti vodilo. Kolikšen del procesorskega časa zdaj zavzemajo operacije, povezane s SCC-jem? Predpostavite enak čas strežbe prekinitve in enako količino sestavljenega prometa kot pri vprašanju v točki **b)**.

d) Predpostavite, da je prenosna hitrost vsakega kanala 64 kb/s. Ali obstaja kakšna možnost uporabe prekinitveno vodenega V/I, kot smo to storili pri vprašanjih v točkah **a)** in **b)**?

e) Ponovite vprašanje **c)** za prenosno hitrost 64 kb/s.

Rešitev:

a) Pri hitrosti 9.600 b/s traja prenos okvirja $\frac{1.064\ \text{b}}{9.600\ \text{b/s}} = 0{,}111$ s. V tem časovnem intervalu mora SCC poslati CPE 128 podatkovnih bajtov, to je $128 \cdot 8 = 1.024$ podatkovnih bitov (40 preostalih bitov okvirja (5 bajtov) pa ne nosi podatkov, temveč krmilno informacijo). Z drugimi besedami, medtem ko se sprejema okvir, bo CPE prekinjana s pogostnostjo $128/0{,}111$ s $= 1.153$-krat na sekundo, kar pomeni enkrat na vsakih 867 μs. Če sta oba kanala sočasno aktivna v obeh smereh, bo CPE v povprečju prekinjana na vsakih $(867\,\mu\text{s})/4 = 217\,\mu$s.

b) V časovnem intervalu 1 s je CPE prekinjena $25 \cdot 128 = 3.200$-krat. Strežba vseh teh prekinitev traja $3.200 \cdot 50\,\mu\text{s} = 160.000\ \mu\text{s} = 0{,}16$ s, kar predstavlja 16 % časa CPE.

c) Število ciklov DMA je enako skupnemu številu podatkovnih bajtov, kar pomeni 3.200 ciklov DMA na sekundo. Vsi prenosi skupaj trajajo $3.200 \cdot 1\,\mu\text{s} = 3.200\,\mu\text{s}$. CPE bo izgubila 20 % tega časa, to je $3.200\,\mu\text{s} \cdot 0{,}2 = 640\,\mu\text{s}$, saj bo vodilo v takšnem deležu zasedeno s prenosi DMA-ja. CPE pa je prekinjena še enkrat na vsak okvir in za to porabi še dodatnih $25 \cdot 50\,\mu\text{s} = 1.250\,\mu\text{s}$. Tako porabi CPE za delo s SCC-jem na sekundo skupno $640\,\mu\text{s} + 1.250\,\mu\text{s} = 1.890\,\mu\text{s}$. To predstavlja samo okrog 0,2 % časa CPE.

d) Prenos enega okvirja zdaj traja $\frac{1.064\ \text{b}}{64.000\ \text{b/s}} = 0{,}0166\,\text{s} = 16.600\,\mu\text{s}$. To ustreza prekinitvi na vsakih $16.600\,\mu\text{s}/128 = 130\,\mu\text{s}$ med sprejemanjem okvirja. Če pa sta oba kanala aktivna sočasno v obeh smereh, CPE tej hitrosti ne bo zmogla slediti.

e) Čeprav se je hitrost prenosa spremenila, je ostalo skupno število podatkovnih bajtov na sekundo enako. Zaradi tega je odgovor na to vprašanje takšen, kot je opisan v točki **c)**.

8 Nabori ukazov CPE

8.1 Prikaz števil v šestnajstiškem zapisu

Prikažite v šestnajstiškem zapisu [1]:

a) pakiran desetiški format za število 23,

b) ASCII znaka 23.

Rešitev:

a) 23

b) 32 33

8.2 Prikaz vrednosti pakiranih desetiških števil

Za vsako izmed naslednjih pakiranih desetiških števil prikažite desetiško vrednost [1]:

a) 0111 0011 0000 1001

b) 0101 1000 0010

c) 0100 1010 0110

Rešitev:

a) 7309

b) 582

c) 0100 1010 0110 ni veljavno pakirano desetiško število, saj je binarna koda druge števke (1010) večja od 1001, zato je v zapisu napaka.

8.3 Najmanjše in največje število v dani predstavitvi

Nek mikroprocesor ima besede dolžine enega bajta. Katero je najmanjše in katero največje celo število, ki je predstavljivo v naslednjih predstavitvah [1]:

a) nepredznačeno,

b) predznačena velikost,

c) eniški komplement,

d) dvojiški komplement,

e) nepredznačeno pakirano desetiško število,

f) predznačeno pakirano desetiško število?

Rešitev:

a) 0; 255

b) -127; 127

c) -127; 127

d) -128; 127

e) 0; 99

f) -9; +9

8.4 Seštevanje pakiranih desetiških števil

Mnogi procesorji zagotavljajo logiko za izvajanje aritmetike na pakiranih desetiških številih. Čeprav so pravila za desetiško aritmetiko podobna tistim za dvojiške operacije, desetiški rezultati zahtevajo nekaj popravkov posameznih števk, če se uporablja dvojiška logika. Razmislite o desetiškem seštevanju dveh nepredznačenih števil. Če je vsako število sestavljeno iz N števk, potem je v vsakem številu 4N bitov. Števili je treba sešteti z binarnim seštevalnikom. Predlagajte preprosto pravilo za popravek rezultata. Na ta način izvedite seštevanje števil 1.698 in 1.798 [1].

Rešitev:
Desetiški števili 1.698 in 1.798 zapišemo z BCD števkami. Če je rezultat seštevanja dveh 4-bitnih BCD števk večji od 9 (dvojiško od 1001), je k rezultatu na trenutnem mestu treba prišteti vrednost 6 (dvojiško 0110), prenos 1 pa prišteti k BCD števki na sosednjem bolj uteženem mestu.

0001	0110	1001	1000
+0001	+0111	+1001	+1000
0010	1101	0010	0000
+ 1	+ 1	+ 1	+0110
0011	1110	0011	0110
	+0110	+0110	
	0100	1001	

Končni rezultat seštevanja je dvakrat podčrtan: 1.698 + 1.798 = 3.496 se v kodu BCD zapiše kot 0011 0100 1001 0110.

8.5 Razlaga ukaza DAA Intelovih mikroprocesorjev x86

Arhitektura Intelovih mikroprocesorjev x86 (Intelova arhitektura, temelječa na procesorju 8086) vključuje tudi ukaz, imenovan DAA (Decimal Adjust after Addition). DAA izvaja naslednje zaporedje ukazov:

```
if ((AL AND 0FH) > 9) OR (AF = 1) then
    AL ← AL + 6;
    AF ← 1;
else
    AF ← 0;
```

```
endif;
if (AL > 9FH) OR (CF = 1) then
    AL ← AL + 60H;
    CF ← 1;
else
    CF ← 0;
endif;
```

Znak H na koncu števila pomeni, da je število zapisano v šestnajstiškem številskem sistemu. AL je 8-bitni register, ki vsebuje rezultat seštevanja dveh nepredznačenih 8-bitnih celih števil. AF je zastavica, ki se postavi, če pride do prenosa iz bita 3 na bit 4 v rezultatu seštevanja. CF je zastavica, ki se postavi, če pride do prenosa iz bita 7 na bit 8. Razložite funkcijo, ki jo opravlja ukaz DAA. [1]

Rešitev:
Ukaz DAA lahko uporabimo za ukazom ADD, da omogočimo uporabo ukaza za seštevanje na dveh 8-bitnih besedah, ki vsebujeta vsaka po dve pakirani desetiški števki. Če pride do desetiškega prenosa (to je takrat, ko je rezultat večji od 9) v desni števki, se ta pokaže bodisi kot števka rezultata, večja od 9, ali postavitev zastavice AF. Če tak prenos obstaja, rezultat popravimo s prištetjem vrednosti 6. Ilustrirajmo to z naslednjim primerom:

$$\begin{array}{r} 27 \\ +46 \\ \hline 6D \\ +06 \\ \hline 73 \end{array}$$

Drugi test podobno popravi prenos iz leve števke v 8-bitnem bajtu. Pakirano desetiško seštevanje nad več števkami lahko sprogramiramo z uporabo navadnega ukaza za seštevanje s prenosom (ADC) v zanki in vstavitvi enega ukaza DAA po vsakem seštevanju.

8.6 Vrednosti zastavic po izvedbi seštevanja za Intel x86

Analizirajte vrednosti zastavic po izvedbi seštevanja za arhitekturo Intel x86 [1].

a) Kakšna bo vrednost zastavic C (Carry), Z (Zero), O (Overflow), S (Sign), P (Parity) in A (Auxiliary Carry), če je bila zadnja izvedena operacija nad 8-bitno besedo (bajtom) na računalniku s procesorjem Intel x86 seštevanje operandov 00000010 in 00000011?

b) Ponovite točko **a)** za seštevanje vrednosti -1 (v dvojiškem komplementu) in +1.

Rešitev:

a)
$$\begin{array}{r} 00000010 \\ +00000011 \\ \hline 00000101 \end{array}$$

Carry = 0; Zero = 0; Overflow = 0; Sign = 0; Parity = 1; Aux. Carry = 0.

Soda pariteta pomeni, da je v rezultatu sodo število enic. Pomožni prenos (Auxiliary Carry) se uporablja pri seštevanju pakiranih desetiških števil. Kadar pride do prenosa iz nižje utežene števke (spodnjega polbajta v zgornji polbajt), se ta zastavica postavi.

b) Število -1 z osmimi biti v dvojiškem komplementu izračunamo tako, da najprej zapišemo število +1 v dvojiškem številskem sistemu kot 00000001. Zdaj izračunamo eniški komplement, kar pomeni, da vse ničle zamenjamo z enicami in vse enice z ničlami. Dobimo 11111110. Nazadnje moramo k temu številu v eniškem komplementu prišteti vrednost 1 po modulu 2^8, kar pomeni, da morebitni prenos iz najbolj uteženega (levega) mesta zavržemo. Tako dobimo število -1, ki ga v dvojiškem komplementu zapišemo kot 11111111.

$$\begin{array}{rl} & 11111111 \\ +&\underline{00000001} \quad (\text{mod } 2^8) \\ \not{1}\ & 00000000 \end{array}$$

Carry = 1; Zero = 1; Overflow = 0; Sign = 0; Parity = 1; Aux. Carry = 1.

Pri seštevanju je prišlo do prenosa iz najbolj uteženega (levega) mesta, ki pa ga zaradi seštevanja po modulu 2^8 zavržemo. Seštevanje desetiških števil -1 in +1 nam v dvojiškem številskem sistemu, kjer je negativno število -1 predstavljeno z dvojiškim komplementom, resnično vrne rezultat 0.

8.7 Vrednosti zastavic po izvedbi odštevanja za Intel x86

Ponovite problem **8.6** za operacijo A − B, kjer A vsebuje desetiško vrednost -16, B pa desetiško vrednost +20 [1].

Rešitev:

Aritmetično logična enota v procesorju namesto odštevanja A − B izvede seštevanje A + (-B). Po zgledu iz problema **8.6** se negativno število A = -16 z 8-bitno besedo v dvojiškem komplementu zapiše kot 11110000, negativno število (-B) pa kot 11101100. Po seštevanju A + (-B) dobimo naslednji rezultat:

$$\begin{array}{rl} & 11110000 \\ +&\underline{11101100} \quad (\text{mod } 2^8) \\ \not{1}\ & 11011100 \end{array}$$

Carry = 1; Zero = 0; Overflow = 0; Sign = 1; Parity = 0; Aux. Carry = 0.

Kakšno je pravilo za zastavico Overflow pri seštevanju dveh predznačenih števil? Če sta obe števili pozitivni ali obe negativni, pride do prekoračitve, če ima rezultat nasprotni predznak. V našem primeru sta obe števili negativni (najbolj uteženi bit je enak 1), negativen je tudi rezultat, zato Overflow = 0. Izračun v desetiškem številskem sistemu: A − B = A + (-B) = -16 + (-20) = -36. Število -36 je v rangu predstavljivih 8-bitnih števil od -128 do +127, zato ni prekoračitve.

8.8 Program za izračun izraza na štirih tipih procesorjev

Primerjajte procesorje, ki imajo ukaze brez naslova, z enim naslovom, dvema naslovoma ali tremi naslovi, tako da napišete program za izračun izraza

$$W = \frac{A + B \times C}{D - E \times F}$$

za vsak navedeni tip procesorja.

V spodnji tabeli so za vsakega od štirih procesorjev navedeni razpoložljivi strojni ukazi, zapisani z mnemoniki, v oklepajih pa je interpretacija teh ukazov.

0 naslovov	**1 naslov**	**2 naslova**	**3 naslovi**
PUSH M (T ← M)	LOAD M (AC←M)	MOVE X,Y (X ← Y)	MOVE X,Y (X ← Y)
POP M (M ← T)	STORE M (M←AC)	ADD X,Y (X ← X + Y)	ADD X,Y,Z (X ← Y + Z)
ADD (T ← (T−1)+T)	ADD M (AC←AC+M)	SUB X,Y (X ← X−Y)	SUB X,Y,Z (X ← Y−Z)
SUB (T ← (T−1)−T)	SUB M (AC←AC−M)	MUL X,Y (X ← X × Y)	MUL X,Y,Z (X ← Y × Z)
MUL (T← (T−1)×T)	MUL M (AC←AC×M)	DIV X,Y (X ← X/Y)	DIV X,Y,Z (X ← Y/Z)
DIV (T← (T−1)/T)	DIV M (AC←AC/M)		

A, B, C, D, E, F, W in M so pomnilniške lokacije, T je element na vrhu sklada, $(T-1)$ element tik pod vrhom sklada, AC je akumulator, spremenljivke X, Y in Z so pomnilniške lokacije ali registri procesorja [1].

Opomba: Pri procesorju, ki ima ukaze z 0 naslovi, sta izjemi ukaza PUSH in POP, ki imata po en naslov, pri procesorju s tremi naslovi pa ukaz MOVE, ki ima dva naslova.

Navodilo: Programe poskusite napisati tako, da boste uporabili čim manj dodatnih začasnih spremenljivk, če sploh katero. V ta namen lahko uporabite tudi W. Vsi operandi morajo ohraniti svoje vrednosti tudi po zaključku programa, končni rezultat pa mora biti v W.

Rešitev:

0 naslovov	**1 naslov**	**2 naslova**	**3 naslovi**
PUSH A	LOAD E	MOVE R0, E	MUL R0, E, F
PUSH B	MUL F	MUL R0, F	SUB R0, D, R0
PUSH C	STORE T	MOVE R1, D	MUL R1, B, C
MUL	LOAD D	SUB R1, R0	ADD R1, A, R1
ADD	SUB T	MOVE R0, B	DIV W, R1, R0
PUSH D	STORE T	MUL R0, C	
PUSH E	LOAD B	ADD R0, A	
PUSH F	MUL C	DIV R0, R1	
MUL	ADD A	MOVE W, R0	
SUB	DIV T		
DIV	STORE W		
POP W			

8.9 Množenje z aritmetičnim in logičnim pomikanjem v levo

Tako aritmetično kot logično pomikanje v levo predstavljata množenje z 2, če ni prekoračitve, če pa pride do prekoračitve, operaciji ustvarita različna rezultata, vendar aritmetično pomikanje v levo ohranja predznak števila. Pokažite, da te izjave veljajo za 5-bitna cela števila v formatu dvojiškega komplementa [1].

Rešitev:

Bitni vzorec	**Vrednost**	**Aritmetično pomikanje v levo**	**Vrednost**	**Logično pomikanje v levo**	**Vrednost**
00000	0	00000	0	00000	0
00001	1	00010	2	00010	2
00010	2	00100	4	00100	4
00011	3	00110	6	00110	6
00100	4	01000	8	01000	8
00101	5	01010	10	01010	10
00110	6	01100	12	01100	12
00111	7	01110	14	01110	14
01000	8	00000	prekoračitev	10000	prekoračitev
01001	9	00010	prekoračitev	10010	prekoračitev
01010	10	00100	prekoračitev	10100	prekoračitev
01011	11	00110	prekoračitev	10110	prekoračitev
01100	12	01000	prekoračitev	11000	prekoračitev
01101	13	01010	prekoračitev	11010	prekoračitev
01110	14	01100	prekoračitev	11100	prekoračitev
01111	15	01110	prekoračitev	11110	prekoračitev
10000	-16	10000	prekoračitev	00000	prekoračitev
10001	-15	10010	prekoračitev	00010	prekoračitev
10010	-14	10100	prekoračitev	00100	prekoračitev
10011	-13	10110	prekoračitev	00110	prekoračitev
10100	-12	11000	prekoračitev	01000	prekoračitev
10101	-11	11010	prekoračitev	01010	prekoračitev
10110	-10	11100	prekoračitev	01100	prekoračitev
10111	-9	11110	prekoračitev	01110	prekoračitev
11000	-8	10000	-16	10000	-16
11001	-7	10010	-14	10010	-14
11010	-6	10100	-12	10100	-12
11011	-5	10110	-10	10110	-10
11100	-4	11000	-8	11000	-8
11101	-3	11010	-6	11010	-6
11110	-2	11100	-4	11100	-4
11111	-1	11110	-2	11110	-2

8.10 Nameni uporabe strojnega ukaza NOP

Nabori ukazov mnogih procesorjev vsebujejo ukaz NOP (No Operation), kar pomeni brez operacije, ki razen inkrementiranja programskega števnika nima nobenega učinka na stanje procesorja. Za katere namene bi ta ukaz lahko bil koristen [1]?

Rešitev:

Strojna koda za ukaz NOP ima dolžino, ki je enaka najkrajši dolžini v naboru ukazov procesorja, in dolžina kateregakoli drugega ukaza je mnogokratnik te dolžine. To dejstvo lahko koristno uporabimo pri razhroščevanju programa. Če želimo iz programa odstraniti nekaj strojnih ukazov, za katere menimo, da so odveč, lahko to naredimo hitro, brez potrebe po spreminjanju programa v urejevalniku, prevajanju in nato ponovnem nalaganju ažurirane strojne kode v pomnilnik kar tako, da v obstoječi strojni kodi programa del kode, ki ga želimo odstraniti, prepišemo s strojno kodo za NOP.

Druga možna uporaba ukaza NOP za razhroščevanje je ta, da v določene testne točke programa vstavimo ukaz NOP, nato pa po potrebi strojno kodo za NOP zamenjamo s strojno kodo za skok v podprogram za razhroščevanje.

Tretja korist ukaza NOP je vnašanje kratkih časovnih zakasnitev z vnaprej znanim časom trajanja (npr. po ukazu za sprožitev AD (analogno-digitalne) pretvorbe vstavimo potrebno število ukazov NOP in s tem vnesemo dovolj zakasnitve, da bo AD pretvorba končana pred izvedbo ukaza za branje vrednosti vzorca).

8.11 Analiza delovanja ukaza CMP pri procesorjih x86

Ukaz CMP (Compare) pri procesorjih x86 odšteje izvorni operand od ciljnega operanda in ažurira statusne zastavice (C, P, A, Z, S, O), vendar ne spremeni nobenega izmed operandov. Ukaz CMP lahko uporabimo, da ugotovimo, ali je ciljni operand večji (>), enak (=) ali manjši (<) od izvornega operanda [1].

a) Predpostavite, da sta operanda interpretirana kot nepredznačeni celi števili. Prikažite, katere statusne zastavice so relevantne za določanje relativne velikosti obeh celih števil in katere vrednosti zastavic ustrezajo pogojnim kodam oziroma relacijam manjši kot, večji kot in enak.

b) Predpostavite, da sta operanda interpretirana kot predznačeni celi števili v dvojiškem komplementu. Prikažite, katere statusne zastavice so relevantne za določanje relativne velikosti obeh celih števil in katere vrednosti zastavic ustrezajo pogojnim kodam oziroma relacijam manjši kot, večji kot in enak.

c) Ukazu CMP lahko sledi ukaz za pogojni skok Jcc (Jump if cc) ali SETcc (Set Condition), kjer je cc sklic na enega izmed 16 pogojev, zbranih v tabeli na naslednji strani. Podrobno razložite, da so pogoji preverjanja za predznačena števila pravilni.

Simbol	Testirani pogoj	Komentar
A, NBE	C = 0 AND Z = 0	Above; Not below or equal (greater than, unsigned)
AE, NB, NC	C = 0	Above or equal; Not below (greater than or equal, unsigned); Not carry
B, NAE, C	C = 1	Below; Not above or equal (less than, unsigned); Carry set
BE, NA	C = 1 OR Z = 1	Below or equal; Not above (less than or equal, unsigned)
E, Z	Z = 1	Equal; Zero (signed or unsigned)
G, NLE	[(S = 1 AND O = 1) OR (S = 0 AND O = 0)] AND [Z = 0]	Greater than; Not less than or equal (signed)
GE, NL	(S = 1 AND O = 1) OR (S = 0 AND O = 0)	Greater than or equal; Not less than (signed)
L, NGE	(S = 1 AND O = 0) OR (S = 0 AND O = 1)	Less than; Not greater than or equal (signed)
LE, NG	(S = 1 AND O = 0) OR (S = 0 AND O = 1) OR (Z = 1)	Less than or equal; Not greater than (signed)
NE, NZ	Z = 0	Not equal; Not zero (signed or unsigned)
NO	O = 0	No overflow
NS	S = 0	Not sign (not negative)
NP, PO	P = 0	Not parity; Parity odd
O	O = 1	Overflow
P	P = 1	Parity; Parity even
S	S = 1	Sign (negative)

Vir: [1]

Rešitev:

a)

Rezultat ukaza CMP	Z	C
cilj < izvor	0	1
cilj > izvor	0	0
cilj = izvor	1	0

b)

Rezultat ukaza CMP	Zastavice
cilj < izvor	S ≠ O
cilj > izvor	S = O AND Z = 0
cilj = izvor	Z = 1

c)
- **Enak:** Operanda sta enaka, zato je rezultat odštevanja enak nič (Z = 1).
- **Večji kot:** Če je A večji kot B in sta A in B oba pozitivna ali oba negativna, operacija (A − B) v dvojiškem komplementu vrne pozitivni rezultat (S = 0) brez prekoračitve (O = 0). Če je A večji kot B in je A pozitiven, B pa negativen, je rezultat bodisi pozitiven (S = 0) brez prekoračitve (O = 0) ali negativen (S = 1) s prekoračitvijo (O = 1). V vseh teh primerih je rezultat različen od nič (Z = 0). Na kratko lahko pogoj zapišemo kot S = O AND Z = 0.
- **Večji ali enak kot:** Razmišljanje je enako kot za pogoj „Večji kot“ s to razliko, da je rezultat lahko bodisi nič bodisi različen od nič, zato v pogoju odpade člen z zastavico Z: S = O.
- **Manjši kot:** Ta pogoj je nasproten od pogoja za „Večji ali enak kot“: S ≠ O.
- **Manjši kot ali enak:** Ta pogoj je nasproten od pogoja za „Večji kot“: S ≠ O OR Z = 1.
- **Ni enak:** Operanda nista enaka, zato rezultat odštevanja ni enak nič: Z = 0.

8.12 Seštevanje šestnajstiških vrednosti z ukazi MMX

Predpostavite, da dva registra vsebujeta naslednji šestnajstiški vrednosti: AB0890C2 in 4598EE50. Kakšen je rezultat njunega seštetja z ukazi MMX (Intelova arhitektura ukazne množice za multimedijo) [1]:

a) za pakirane bajte,

b) za pakirane besede?

Rešitev:

a) Seštevanje po bajtih:

```
  AB 08 90 C2
+ 45 98 EE 50
  F0 A0 7E 12
```

b) Seštevanje po 16-bitnih besedah:

```
  AB08  90C2
+ 4598  EE50
  F0A0  7F12
```

8.13 Pretvorba iz obratne poljske notacije v infiksno

Pretvorite naslednje formule iz obratne poljske notacije v infiksno [1].

a) A B + C + D ×

b) A B / C D / +

c) A B C D E + × × /

d) A B C D E + F / + G − H / × +

Rešitev:

a) (A + B + C) × D

b) (A / B) + (C / D)

c) A / (B × C × (D + E))

d) A + (B × ((C + (D + E) / F) − G) / H)

8.14 Pretvorba iz infiksne notacije v obratno poljsko

Pretvorite naslednje formule iz infiksne notacije v obratno poljsko notacijo [1].

a) A + B + C + D + E

b) (A + B) × (C + D) + E

c) (A × B) + (C × D) + E

d) (A − B) × (((C − D × E) / F) / G) × H

Rešitev:

a) A B + C + D + E +

b) A B + C D + × E +

c) A B × C D × + E +

d) A B − C D E × − F / G / × H ×

8.15 Uporaba ukaza IMUL za procesorje Intel x86

Nabor ukazov Intelove računalniške arhitekture x86 vsebuje tudi ukaz

```
imul op1, op2, takojšnji-operand
```

Ta ukaz pomnoži op2, ki je lahko bodisi vrednost v registru ali v pomnilniku, z vrednostjo takojšnjega operanda in postavi rezultat v op1, ki mora biti register. V naboru ukazov ni drugih ukazov take vrste s tremi operandi. Katera je možna uporaba tega ukaza [1]?

Rešitev:
To je primer namenskega ukaza CISC, načrtovanega za podporo prevajalnika. Zamislimo si primer indeksiranja polja, kjer so elementi polja dolgi 32 bitov. Naslednji ukaz je ravno to, kar potrebujemo:

```
imul ebx, i, 32
```

ebx je 32-bitni register, ki zdaj vsebuje odmik v bajtih v polje z indeksom i.

8.16 Ureditev bajtov v pomnilniku

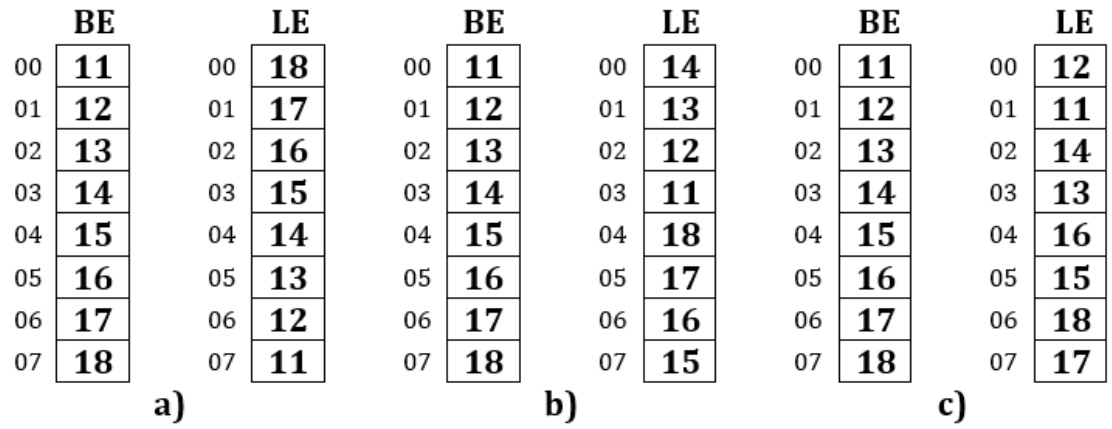
Za naslednje podatkovne strukture narišite ureditev bajtov v pomnilniku za „veliki endian“ in „mali endian“ [1].

a)
```
struct {
    double i; //0x1112131415161718
}s1;
```

b)
```
struct {
    int i; //0x11121314
    int j; //0x15161718
}s2;
```

c)
```
struct {
    short i; //0x1112
    short j; //0x1314
    short k; //0x1516
    short l; //0x1718
}s3;
```

Rešitev:
V rešitvi je pomen kratic naslednji:
BE: veliki endian („big endian“)
LE: mali endian („little endian“)



8.17 Program za ugotavljanje ureditve bajtov*

Napišite kratek program, ki bo ugotovil, ali računalnik uporablja ureditev bajtov „veliki endian“ ali „mali endian“. Program poženite na svojem računalniku in zapišite izhod iz programa [1].

Rešitev:
Eden izmed možnih programov za rešitev problema je napisan spodaj:

```
#include <stdio.h>
main()
{
   int integer;
   char *p;

   integer = 0x30313233; /* ASCII kode za znake '0', '1', '2', '3' */
   p = (char *)&integer;

   if (*p=='0' && *(p+1)=='1' && *(p+2)=='2' && *(p+3)=='3')
      printf("To je racunalnik veliki endian.\n");
   else if (*p=='3' && *(p+1)=='2' && *(p+2)=='1' && *(p+3)=='0')
      printf("To je racunalnik mali endian.\n");
   else
      printf("Napaka v logiki za dolocitev, ali je racunalnik veliki ali mali endian.\n");
}
```

Ta program smo pognali na računalniku s procesorjem Intel Core i7. Program je izpisal naslednje besedilo: To je racunalnik mali endian.

9 Načini naslavljanja in formati ukazov CPE

9.1 Razumevanje različnih načinov naslavljanja (1/4)

Podanih je nekaj vrednosti v pomnilniku računalnika:

- Beseda 20 vsebuje vrednost 40.
- Beseda 30 vsebuje vrednost 50.
- Beseda 40 vsebuje vrednost 60.
- Beseda 50 vsebuje vrednost 70.

Računalnik ima procesor z akumulatorjem in ukaze z enim naslovom. Katere vrednosti naslednji ukazi naložijo v akumulator [1]?

a) LOAD IMMEDIATE 20

b) LOAD DIRECT 20

c) LOAD INDIRECT 20

d) LOAD IMMEDIATE 30

e) LOAD DIRECT 30

f) LODE INDIRECT 30

Rešitev:

a) 20

b) 40

c) 60

d) 30

e) 50

f) 70

9.2 Razumevanje različnih načinov naslavljanja (2/4)

Naslov v programskem števniku označimo z X1. Ukaz, shranjen na naslovu X1, ima naslovni del (sklic na operand) X2. Operand, ki je potreben za izvedbo ukaza, je shranjen v pomnilniško besedo na naslovu X3. Indeksni register vsebuje vrednost X4. Kakšna je zveza med temi različnimi količinami za naslednje načine naslavljanja ukaza [1]?

a) neposredno

b) posredno

c) relativno glede na PC

d) indeksno

Rešitev:

a) X3 = X2

b) X3 = (X2)

c) X3 = X1 + X2 + 1

d) X3 = X2 + X4

9.3 Razumevanje različnih načinov naslavljanja (3/4)

Naslovno polje v ukazu vsebuje desetiško vrednost 14. Kje se nahaja ustrezni operand za

a) sprotno naslavljanje,

b) neposredno naslavljanje,

c) posredno naslavljanje,

d) registrsko naslavljanje,

e) registrsko posredno naslavljanje [1]?

Rešitev:
Operand se nahaja

a) v naslovnem polju (njegova vrednost je 14).

b) na pomnilniški lokaciji z naslovom 14.

c) na pomnilniški lokaciji, katere naslov se nahaja na pomnilniški lokaciji 14.

d) v registru 14.

e) na pomnilniški lokaciji, katere naslov je v registru 14.

9.4 Razumevanje različnih načinov naslavljanja (4/4)

Predpostavite 16-bitni procesor z naslednjo vsebino pomnilnika od lokacije 200 naprej:

200	*naloži v AC*	*način*
201	500	
202	*naslednji ukaz*	

Polje *naloži v AC* v prvi besedi (operacijska koda) označuje, da ta ukaz naloži vrednost v akumulator. Polje *način* specificira način naslavljanja in, če je relevantno, označuje izvorni register. Predpostavite, da je izvorni register, če je uporabljen, register R1, ki ima vrednost 400. Procesor ima tudi bazni register, ki vsebuje vrednost 100. Vrednost 500 na lokaciji 201 je lahko del pri izračunu naslova. Predpostavite, da lokacija 399 vsebuje vrednost 999, lokacija 400 vsebuje vrednost 1.000 in tako naprej. Določite dejanski naslov in operand, ki se bo naložil za naslednje načine naslavljanja [1].

a) neposredno

b) takojšnje

c) posredno

d) relativno glede na PC

e) s premikom

f) registrsko

g) registrsko posredno

h) avtoindeksno z inkrementom in uporabo R1

Rešitev:

	dejanski naslov	**operand**
a)	500	1.100
b)	201	500
c)	1.100	1.700
d)	702	1.302
e)	600	1.200
f)	R1	400
g)	400	1.000
h)	400	1.000

9.5 Ukaz za vejanje z relativnim naslavljanjem (1/2)

Ukaz za vejanje z relativnim naslavljanjem glede na programski števnik je dolg 3 bajte. Desetiški naslov ukaza je 256.028. Določite naslov cilja vejanja, če je predznačeni odmik ukaza -31 [1].

Rešitev:
Spomnimo se, da relativno naslavljanje uporablja vsebino programskega števnika, ki kaže na naslednji ukaz za trenutnim ukazom. V tem primeru je trenutni ukaz na desetiškem naslovu 256.028 in je dolg 3 bajte, zato programski števnik vsebuje vrednost 256.031. Glede na odmik -31 je dejanski naslov 256.031 -31 = 256.000.

9.6 Ukaz za vejanje z relativnim naslavljanjem (2/2)

Ukaz za vejanje z relativnim naslavljanjem glede na programski števnik je shranjen v pomnilniku na naslovu 620_{10}. Vejanje se izvede na naslov 530_{10}. Naslovno polje v ukazu je dolgo 10 bitov. Koliko znaša dvojiška vrednost v tem polju [1]?

Rešitev:
(PC + 1) + relativni naslov = dejanski naslov
relativni naslov = dejanski naslov − (PC + 1)
relativni naslov = 530 − (620 + 1) = -91

S pretvorbo desetiškega števila -91 v 10-bitno število v dvojiškem komplementu dobimo za odmik v naslovnem polju vrednost 1110100101.

9.7 Ukaz s posrednim načinom naslavljanja

Kolikokrat mora procesor dostopiti do pomnilnika, ko včita in izvaja ukaz s posrednim načinom naslavljanja, če je ukaz

a) računska operacija, ki zahteva en operand,

b) vejanje [1]?

Rešitev:

a) Procesor mora dostopiti do pomnilnika trikrat − včitanje ukaza, včitanje naslova operanda in včitanje samega operanda.

b) Procesor mora dostopiti do pomnilnika dvakrat − včitanje ukaza, včitanje naslova operanda in njegova naložitev v programski števnik.

9.8 Bazno indeksni način naslavljanja

Nek procesor pozna tudi bazno indeksni način naslavljanja. Predpostavite, da je izvajanje prišlo do ukaza s takšnim načinom naslavljanja in da ukaz specificira odmik 1.970. Trenutna vrednost baznega registra je 48.022, indeksnega registra pa 8. Kolikšen je naslov operanda [1]?

Rešitev:
Procesor izračuna naslov operanda s seštevanjem odmika, vsebine baznega registra in vsebine indeksnega registra. Torej je operandov pomnilniški naslov enak 1.970 + 48.022 + 8 = 50.000.

9.9 Naslavljanje s predzmanjšanjem in postpovečanjem

Definirajmo naslednje zapise. EA = (X)+ je dejanski naslov, ki je enak vsebini lokacije X, X pa se inkrementira za dolžino ene besede po izračunu dejanskega naslova. EA = -(X) je dejanski naslov, ki je enak vsebini lokacije X, pri čemer se X dekrementira za dolžino ene besede, preden se izračuna dejanski naslov. EA = (X)- je dejanski naslov, ki je enak vsebini lokacije X, kjer se X dekrementira za dolžino ene besede po izračunu dejanskega naslova. Predpostavite spodaj navedene ukaze,

če je vsak izmed njih zapisan v formatu, kjer je za operacijo najprej zapisan izvorni operand, nato ciljni operand, rezultat operacije pa se postavi v ciljni operand.

a) OP X, (X)

b) OP (X), (X)+

c) OP (X)+, (X)

d) OP -(X), (X)

e) OP -(X), (X)+

f) OP (X)+, (X)+

g) OP (X)-, (X)

Če uporabimo X za kazalec sklada, kateri izmed teh ukazov lahko sname dva vrhnja elementa s sklada, izvede določeno operacijo (npr. sešteje izvor k cilju in shrani rezultat v cilj) in postavi rezultat nazaj na sklad? Za vsak takšen ukaz povejte, ali sklad narašča proti nižjim ali višjim pomnilniškim naslovom [1].

Rešitev:

a) Ne, ker je izvorni operand vsebina X-a, ne pa vrha sklada, ki je na lokaciji, kamor kaže X.

b) Ne, ker se včita dvakrat isti operand.

c) Da. Sklad raste proti višjim pomnilniškim naslovom.

d) Ne, ker se včita dvakrat isti operand.

e) Ne, ker se včita dvakrat isti operand.

f) Ne, ker se kazalec sklada inkrementira dvakrat, tako da se rezultat zavrže.

g) Da. Sklad raste proti nižjim pomnilniškim naslovom.

9.10 Skladovno naslavljanje

Predpostavite skladovno usmerjeni procesor, ki vsebuje skladovni operaciji PUSH in POP. Aritmetične operacije avtomatično zahtevajo vrhnji element ali dva vrhnja elementa sklada. Začnite s praznim skladom. Prikažite elemente na skladu med izvajanjem naslednjih ukazov [1].

```
PUSH 4
PUSH 7
PUSH 8
ADD
PUSH 10
SUB
MUL
```

Rešitev:

Ukaz	**Sklad (vrh na levi)**
PUSH 4	4
PUSH 7	7, 4
PUSH 8	8, 7, 4
ADD	15, 4
PUSH 10	10, 15, 4
SUB	5, 4
MUL	20

9.11 Izračun največjega števila enooperandnih ukazov*

Predpostavite ukazno množico, ki uporablja ukaze dolžine 16 bitov. Operandovska polja so dolga 6 bitov. Obstaja K dvooperandovskih ukazov in L ukazov brez operanda. Kolikšno je največje število enooperandnih ukazov, ki so lahko podprti [1]?

Rešitev:

Naj bo X število ukazov z enim naslovom (z enim operandom). Možnost, da imamo K dvonaslovnih, X enonaslovnih in L breznaslovnih ukazov v 16-bitni ukazni besedi, zahteva, da velja:

$$2^6 \cdot 2^6 \cdot K + 2^6 \cdot X + L = 2^{16}$$

Iz enačbe izrazimo X:

$$X = \frac{2^{16} - 2^{12} \cdot K - L}{2^6}$$

Za preveritev tega rezultata predpostavimo primer, da nimamo ukazov brez naslova in z dvema naslovoma, to pomeni, da je $L = K = 0$. V tem primeru dobimo:

$$X = \frac{2^{16}}{2^6} = 2^{10}$$

Rezultat je pravilen, saj ob enem 6-bitnem naslovnem polju ostane 10 bitov za polje z operacijsko kodo, ki lahko zavzame 2^{10} kombinacij.

9.12 Načrtovanje operacijske kode spremenljive dolžine*

Načrtajte operacijsko kodo spremenljive dolžine, ki omogoča, da se vse navedeno zakodira v 36-bitni ukaz:

- 7 ukazov z dvema 15-bitnima naslovoma in eno 3-bitno številko registra,
- 500 ukazov z enim 15-bitnim naslovom in eno 3-bitno številko registra,
- 50 ukazov brez naslovov ali registrov [1].

Rešitev:

36-bitni ukaz razdelimo na štiri polja: A, B, C in D. Polje A so prvi trije biti, polje B je naslednjih 15 bitov, polje C je naslednjih 15 bitov in polje D so zadnji trije biti. 7 ukazov s tremi operandi uporablja polja B, C in D za operande in A za operacijsko kodo. Naj bodo kombinacije 000 do 110 operacijske kode in kombinacija 111 koda, ki označuje, da ima ukaz manj kot tri operande. 500 ukazov z dvema operandoma je specificiranih s kombinacijo 111 v polju A in operacijsko kodo v polju B z operandoma v C in D. V polju B porabimo za 500 ukazov samo 9 od 15 bitov ($2^9 = 512 > 500$). Operacijske kode za 50 ukazov brez operandov lahko zakodiramo s preostalimi šestimi biti v polju B ($2^6 = 64 > 50$).

9.13 Izračun dolžine programske kode za izračun izraza

Izhajajte iz rezultatov naloge **8.8**. Predpostavite, da so pomnilniški naslovi 16-bitni. Sklici na operande v ukazih so 16-bitni, če se nanašajo na pomnilniški naslov, in 4-bitni, če se nanašajo na register. Procesor z enim naslovom uporablja akumulator, procesorja z dvema ali tremi naslovi pa imata 16 registrov in ukaze, ki delujejo nad vsemi kombinacijami pomnilniških naslovov in registrov. Predpostavite 8-bitno operacijsko kodo in dolžine ukazov, ki so večkratniki 4 bitov. Koliko bitov programa potrebuje vsak izmed procesorjev, da izračuna rezultat W [1]?

Rešitev:

a) Format ukaza brez naslova je sestavljen iz 8-bitne operacijske kode in izbirnega 16-bitnega naslova (operaciji PUSH M in POP M). Program ima 12 ukazov, 7 med njimi ima sklic na naslov. Skupno potrebno število bitov za program je:

$N_0 = 12\times8 + 7\times16 = 208$ bitov.

b) Format ukaza z enim naslovom je sestavljen iz 8-bitne operacijske kode in 16-bitnega naslova. Program ima 11 ukazov. Skupno potrebno število bitov za program je:

$N_1 = 11\times(8 + 16) = 264$ bitov.

c) Format ukaza z dvema naslovoma ima 8-bitno operacijsko kodo in sklica na dva operanda. Za sklic na operand v registru se uporabijo 4 biti, za sklic na operand v pomnilniku pa 16 bitov. Program ima 9 ukazov, v katerih je 7 sklicev na operand v pomnilniku in 11 sklicev na operand v registru. Skupno potrebno število bitov za program je:

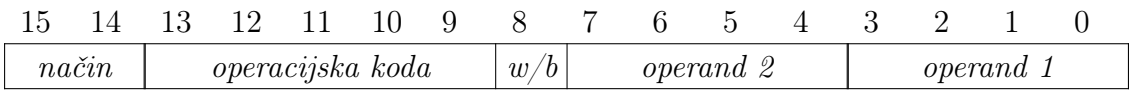
$N_2 = 9\times8 + 7\times16 + 11\times4 = 228$ bitov.

d) Format ukaza s tremi naslovi je sestavljen iz 8-bitne operacijske kode in sklicev na tri operande, med katerimi je en operand v registru in dva operanda v pomnilniku ali pa sta dva v registrih in eden v pomnilniku. Program je sestavljen iz 5 ukazov, v katerih je skupaj 7 sklicev na pomnilnik in 8 na registre. Skupno potrebno število bitov za program je:

$N_3 = 5\times8 + 7\times16 + 8\times4 = 184$ bitov.

9.14 Število operacijskih kod za mikroprocesor Zilog Z8001

16-bitni mikroprocesor Zilog Z8001 ima naslednji splošni format ukazov:



Polje *način* specificira, kako iz operandovskih polj locirati operande. Bit *w/b* se uporablja v nekaterih ukazih za specifikacijo o tem, ali sta operanda bajta ali 16-bitni besedi. Polje *operand 1* lahko (odvisno od vsebine polja *način*) specificira enega izmed 16 splošnonamenskih registrov. Polje *operand 2* lahko specificira katerikoli splošnonamenski register razen registra R0. Kadar so v polju *operand 2* same ničle, vsaka izmed originalnih operacijskih kod dobi nov pomen [1].

a) Koliko operacijskih kod ima mikroprocesor Zilog Z8001?

b) Predlagajte učinkovit način za preskrbo več operacijskih kod in komentirajte posledice.

Rešitev:

a) 5-bitno polje za operacijsko kodo lahko zavzame $2^5 = 32$ različnih vrednosti. Vsako vrednost v tem polju lahko interpretiramo na dva različna načina v odvisnosti od tega, ali so v polju *operand 2* same ničle ali ne. Število predstavljivih operacijskih kod se zato podvoji, tako da lahko zakodiramo 64 različnih ukazov.

b) Lahko bi pridobili dodatnih 32 operacijskih kod, če bi v ta namen dodelili še en bitni vzorec v polju *operand 2*. Na primer, za pridobitev več operacijskih kod bi lahko uporabili vzorec 0001. Cena za to bi bila omejena fleksibilnost pri programiranju, saj zdaj v polju *operand 2* ne bi mogli več specificirati tudi registra R1.

10 Razvoj programov v zbirnem jeziku

10.1 Določanje vrednosti zastavice C po izvedbi programa

Odgovorite na spodnji vprašanji, če predpostavite zbirnik za procesorje Intelove računalniške arhitekture x86 [1].

a) Kakšna je vrednost statusne zastavice C (carry) po izvedbi naslednjih ukazov:

```
mov al, 3
add al,4
```

b) Kakšna je vrednost statusne zastavice C (carry) po izvedbi naslednjih ukazov:

```
mov al, 3
sub al, 4
```

Rešitev:

a) Zastavica C ima vrednost 0.

b) Zastavica C ima vrednost 1.

10.2 Vrednost zastavic Z, S in O po izvedbi ukaza cmp

Predpostavite naslednji ukaz v zbirniku NASM (Netwide Assembler for the x86) za procesorje Intelove računalniške arhitekture x86:

```
cmp vleft, vright
```

Za predznačena števila so pomembne tri statusne zastavice. Če je vleft = vright, se postavi zastavica Z (zero). Če je vleft > vright, se zastavica Z zbriše in zastavica S (sign) ima enako vrednost kot zastavica O (overflow). Če je vleft < vright, se zastavica Z zbriše, zastavici S in O pa imata različni vrednosti [1].

a) Razložite, zakaj velja S = O, če vleft > vright? Delovanje statusnih zastavic Z, S in O ilustrirajte s primeri štiribitnih predznačenih števil z zapisom v dvojiškem komplementu.

b) Razložite, zakaj velja S ≠ O, če vleft < vright? Delovanje statusnih zastavic Z, S in O ilustrirajte s primeri štiribitnih predznačenih števil z zapisom v dvojiškem komplementu.

Rešitev:

a) vleft > vright

Če ni prekoračitve, ima razlika vleft − vright pravilno pozitivno vrednost, zato je S = O = 0. Če pa pride do prekoračitve, razlika nima pravilne pozitivne vrednosti, ampak je negativna, zato je S = O = 1. Skratka, pri pogoju vleft > vright po operaciji odštevanja vleft − vright za zastavici S in O velja zveza S = O.

- Primer 1: vleft = 6, vright = 4, torej 6 − 4 = 6 + (-4) = 2

  0110 = 6
  +1100 = -4
  0010 = 2, S = O = 0

- Primer 2: vleft = 6, vright = -4, torej 6 − (-4) = 6 + 4 = prekoračitev

  0110 = 6
  +0100 = 4
  1010 = prekoračitev, S = O = 1

b) vleft < vright

Če ni prekoračitve, ima razlika vleft − vright pravilno negativno vrednost, zato je O = 0 in S = 1. Če pa pride do prekoračitve, razlika nima pravilne negativne vrednosti, ampak je pozitivna, zato je O = 1 in S = 0. Skratka, pri pogoju vleft < vright po operaciji odštevanja vleft − vright za zastavici S in O velja zveza S ≠ O.

- Primer 1: vleft = 4, vright = 6, torej 4 − 6 = 4 + (-6) = -2

  0100 = 4
  +1010 = -6
  1110 = -2, S = 1, O = 0

- Primer 2: vleft = -6, vright = 4, torej -6 − 4 = -6 + (-4) = prekoračitev

  1010 = -6
  +1100 = -4
  0110 = prekoračitev, S = 0, O = 1

10.3 Programiranje v zbirniku za procesorje x86 (1/9)

Predpostavite naslednji odsek kode v zbirniku NASM (Netwide Assembler for the x86) za procesorje Intelove računalniške arhitekture x86:

```
mov al, 0
cmp al, al
je next
```

Napišite ekvivalentni program, ki bo imel en sam ukaz [1].

Rešitev:
V register al se naloži vrednost 0, v ukazu cmp pa se glede na izraz 0 − 0 ažurirajo vrednosti zastavic. Zastavica Z se postavi. V ukazu je je pogoj (Z=1) izpolnjen, zato se izvede skok na naslov next. Vse tri ukaze lahko zamenjamo z enim samim ukazom, ki izvede brezpogojni skok na naslov next:

```
jmp next
```

10.4 Programiranje v zbirniku za procesorje x86 (2/9)

Za inicializacijo podatkov lahko uporabimo smernice, ki inicializirajo več lokacij. Na primer

```
db 0x55, 0x56, 0x57
```

rezervira tri bajte in inicializira njihove vrednosti.

NASM podpira posebni žeton $ za izračune, ki vključujejo trenutno mesto v zbirniškem programu. To pomeni, da $ izračuna mesto v zbirniškem programu na začetku vrstice, ki vsebuje izraz. Ob upoštevanju teh dejstev razmislite o naslednjem zaporedju smernic:

```
message db 'Zdravo, svet!'
msglen equ $-message
```

Katera vrednost se dodeli simbolu msglen [1]?

Rešitev:
Simbolu msglen se dodeli konstantna vrednost 13.

10.5 Programiranje v zbirniku za procesorje x86 (3/9)

Predpostavite, da tri simbolične spremenljivke V1, V2 in V3 vsebujejo nepredznačene celoštevilčne vrednosti. V zbirniku NASM za procesorje Intelove računalniške arhitekture x86 napišite del kode, ki bo premaknil najmanjšo vrednost v register ax. Uporabljate lahko samo ukaze mov, cmp in jbe [1].

Rešitev:

```
V1: resw 1      ; vrednosti morajo biti prirejene
V2: resw 1      ; pred zagonom programa
V3: resw 1

main: mov ax, V1      ; naloži V1 za testiranje
      cmp ax, V2      ; če ax <= V2, potem
      jbe L1          ; skok na L1
      mov ax, V2      ; sicer premakni V1 v ax
L1:   cmp ax, V3      ; če ax <= V2, potem
      jbe L2          ; skok na L2
      mov ax, V3      ; sicer premakni V1 v ax
L2:
```

10.6 Programiranje v zbirniku za procesorje x86 (4/9)

Za medsebojno zamenjavo vsebine dveh registrov lahko uporabimo ukaz xchg. Predpostavimo, da nabor ukazov za procesorje Intelove računalniške arhitekture x86 ne bi podpiral tega ukaza [1].

a) Izvedite ukaz xchg ax, bx samo z uporabo ukazov push in pop.

b) Izvedite ukaz `xchg ax, bx` samo z uporabo ukaza `xor` (ne uporabljajte drugih registrov).

Rešitev:

a)
```
push ax
push bx
pop ax
pop bx
```

b)
```
xor ax, bx
xor bx, ax
xor ax, bx
```

10.7 Programiranje v zbirniku za procesorje x86 (5/9)

V naslednjem programu, napisanem v zbirniku NASM za procesorje Intelove računalniške arhitekture x86, predpostavite, da so `a`, `b`, `x` in `y` simboli za lokacije glavnega pomnilnika. Ugotovite, kaj ta program dela [1].

```
        mov eax, a
        mov ebx, b
        xor eax, x
        xor ebx, y
        jnz L2
L1:             ; zaporedje ukazov A
        jmp L3
L2:             ; zaporedje ukazov B
L3:
```

Rešitev:
Če je `x = a` in hkrati `y = b`, izvedi zaporedje ukazov A, sicer izvedi zaporedje ukazov B.

10.8 Programiranje v zbirniku za procesorje x86 (6/9)

Če mikroprocesorji Intel delujejo v realnem načinu, lahko naslovijo le prvi 1 MB naslovnega prostora. Mikroprocesorja 8086 in 8088 delujeta samo v realnem načinu, kasnejši mikroprocesorji pa v realnem ali zaščitenem načinu. Delovanje v realnem načinu omogoča aplikacijski programski opremi, napisani za procesorja 8086/8088, da deluje tudi s procesorjem 80286 in naslednjimi brez sprememb v programski opremi. Vsi ti mikroprocesorji ob vklopu napajanja po privzetem začnejo delovati v realnem načinu. V realnem načinu se pomnilnik doseže s kombinacijo segmentnega naslova in naslova odmika. Segmentni naslov, ki se nahaja v enem izmed segmentnih registrov, definira začetni naslov kateregakoli pomnilniškega segmenta velikosti 64 KB. Naslov odmika izbere poljubno lokacijo znotraj 64 KB dolgega segmenta v pomnilniku.

Mikroprocesorji 8086-80286 imajo štiri različne 16-bitne segmentne registre (cs – kodni segment, ss – skladovni segment, ds – podatkovni segment in es – dodatni segment) in naslov odmika (ip za naslov ukaza, sp ali bp za naslov sklada, bx, di, si, 8-bitno število ali 16-bitno število za naslov podatka in di za ciljni naslov niza). V realnem načinu se vsakemu segmentnemu registru interno z desne strani pripne 0H. Znak H pomeni, da je to šestnajstiška ničla, sestavljena iz štirih bitov 0. Dejansko to pomeni, da se vsebina segmentnega registra pomnoži s 16. To tvori 20-bitni naslov, ki predstavlja začetni naslov znotraj prvega 1 MB pomnilnika. Odmik do 64 KB od začetka segmenta določi naslov odmika.

Poiščite pomnilniški naslov naslednjega ukaza, ki ga bo izvedel mikroprocesor Intel x86, če ta dela v realnem načinu za naslednje kombinacije v registrih cs:ip. Pri tej nalogi je segmentni register cs, naslov odmika pa vsebuje programski števnik ip [4].

a) cs = 1000H in ip = 2000H

b) cs = 2000H in ip = 1000H

c) cs = 2300H in ip = 1A00H

d) cs = 1A00H in ip = B000H

e) cs = 3456H in ip = ABCDH

Rešitev:

a) 12000H

b) 21000H

c) 24A00H

d) 25000H

e) 3F12DH

10.9 Programiranje v zbirniku za procesorje x86 (7/9)

Določite pomnilniško lokacijo, naslovljeno z naslednjimi kombinacijami registrov mikroprocesorja Intel 80286, ki deluje v realnem načinu [4].

a) ds = 1000H in di = 2000H

b) ds = 2000H in si = 1002H

c) ss = 2300H in bp = 3200H

d) ds = A000H in bx = 1000H

e) ss = 2900H in sp = 3A00H

Rešitev:
Delovanje mikroprocesorjev Intel x86 v realnem načinu je predstavljeno v nalogi **10.8**. Pri tej nalogi se za določitev začetka segmenta uporabljata segmentna registra ds ali ss, za naslov odmika pa registri di, si, bp, bx ali sp.

a) 12000H

b) 21002H

c) 26200H

d) A1000H

e) 2CA00H

10.10 Programiranje v zbirniku za procesorje x86 (8/9)

Predpostavite, da imajo v mikroprocesorju Intelove računalniške arhitekture x86 navedeni registri in odmik list naslednje vrednosti: ds =1100H, bx = 0200H, list = 0250H in si = 0500H. Določite naslov, do katerega dostopa vsak izmed naslednjih ukazov, če deluje mikroprocesor v realnem načinu [4].

a) mov list[si],edx

b) mov cl,list[bx+si]

c) mov ch,[bx+si]

Rešitev:
Za dostop do pomnilnika se uporablja podatkovni segmentni register ds in odmik, specificiran v ukazu. Delovanje mikroprocesorjev Intel x86 v realnem načinu je predstavljeno v nalogi **10.8**.

a) 11750H

b) 11950H

c) 11700H

10.11 Programiranje v zbirniku za procesorje x86 (9/9)

Razvijte zaporedje ukazov v zbirniku za mikroprocesorje Intelove računalniške arhitekture x86, ki seštejejo 8-bitno BCD število v registrih ax in bx k 8-bitnemu BCD številu v registrih cx in dx. ax in cx sta najbolj utežena registra. Po seštevanju mora biti rezultat v registrih cx in dx [4].

Rešitev:

```
push    ax          ; shranemo originalno vsebino registra ax
mov     al, bl
add     al, dl      ; bl + dl
daa                 ; desetiška prilagoditev po seštevanju bl + dl
mov     dl, al      ; shranemo BCD števko 0 rezultata v dl
mov     al, bh
adc     al, dh      ; bh + dh
```

```
daa             ; desetiška prilagoditev po seštevanju bh + dh
mov   dh, al    ; shranemo BCD števko 1 rezultata v dh
pop   ax        ; povrnemo originalno vsebino registra ax
adc   al, cl    ; al + cl
daa             ; desetiška prilagoditev po seštevanju al + cl
mov   cl, al    ; shranemo BCD števko 2 rezultata v cl
mov   al, ah
adc   al, ch    ; ah + ch
daa             ; desetiška prilagoditev po seštevanju ah + ch
mov   ch, al    ; shranemo BCD števko 3 rezultata v ch
```

10.12 Programiranje v zbirniku za procesorje ARM (1/8)

Nek mikroprocesor iz družine ARM ima v registru r0 vrednost r0 = 0x20008000.

Vsebina v pomnilniku je takšna:

Naslov pomnilnika	**Vsebina**
0x20008007	0x79
0x20008006	0xCD
0x20008005	0xA3
0x20008004	0xFD
0x20008003	0x0D
0x20008002	0xEB
0x20008001	0x2C
0x20008000	0x1A

Kakšna je vrednost registrov r1 in r0 po izvedbi naslednjih ukazov? Predpostavimo, da se za organizacijo podatkov v pomnilniku uporablja mali endian. Ti ukazi se izvajajo neodvisno, tj. niso del programa.

a) `ldrsb r1, [r0, #2]`

b) `ldrh r1, [r0], #2`

c) `ldrsh r1, [r0, #4]!`

Rešitev:

a) `ldrsb` (Load Register Signed Byte) naloži bajt iz pomnilnika in mu razširi predznak na 32 bitov.
r1 = 0xFFFFFFEB, r0 = 0x20008000

b) `ldrh` (Load Register Halfword) naloži polbesedo iz pomnilnika in ga razširi z ničlami do 32 bitov.
r1 = 0x00002C1A, r0 = 0x20008002

c) `ldrsh` (Load Register Signed Halfword) naloži polbesedo iz pomnilnika in mu razširi predznak na 32 bitov.
r1 = 0xFFFFA3FD, r0 = 0x20008004

10.13 Programiranje v zbirniku za procesorje ARM (2/8)

Nek mikroprocesor iz družine ARM ima naslednji vrednosti v registrih r0 in r1:
r0 = 0x20000000 in
r1 = 0x12345678.

Vsi bajti v pomnilniku so inicializirani na 0x00. Recimo, da je bil naslednji zbirniški program uspešno izveden. Narišite tabelo, ki prikazuje vrednost pomnilnika, če procesor uporablja mali endian.

```
str r1, [r0], #4
str r1, [r0, #4]!
str r1, [r0, #4]
```

Rešitev:

Naslov pomnilnika	Vsebina
0x20000000	0x78
0x20000001	0x56
0x20000002	0x34
0x20000003	0x12
0x20000004	0x00
0x20000005	0x00
0x20000006	0x00
0x20000007	0x00
0x20000008	0x78
0x20000009	0x56
0x2000000A	0x34
0x2000000B	0x12
0x2000000C	0x78
0x2000000D	0x56
0x2000000E	0x34
0x2000000F	0x12

10.14 Programiranje v zbirniku za procesorje ARM (3/8)

V zbirniški datoteki main.s za procesor ARM Cortex-M4 s sintakso GNU napišite program, ki bo izračunal dolžino niza Lep pozdrav!\n. Znaka \null na koncu niza ne prištevajte k dolžini. Končni rezultat (dolžina niza) naj bo shranjen v registru r2.

Rešitev:

Program v zbirniku (main.s)

```
.cpu cortex-m4
.syntax unified

    .section .text
    .global main

main:
    ldr r0,=niz     // r0 kaže na začetek niza
    mov r2,#0       // inicializacija dolžine

loop:
    ldrb r1,[r0],#1 // branje naslednjega znaka niza, r0 = r0 + 1
    cbz r1, exit    // vejanje, če je znak enak null
    add r2,#1       // inkrementiranje števila znakov v nizu
    b  loop         // ponovitev zanke

exit:
    b exit          // neskončna zanka

    .section .data

niz:
    .asciz "Lep pozdrav!\n" // niz zaključen z null

    .end
```

Ko program pride v neskončno zanko na oznaki exit:, bo register r2 vseboval vrednost 13, ki predstavlja dolžino niza (vključno z znakom za prehod v novo vrstico).

10.15 Programiranje v zbirniku za procesorje ARM (4/8)

Podprogram v zbirniku za mikroprocesorje ARM lahko sprejme do štiri argumente prek registrov r0-r3. Če podprogram sprejme več kot štiri argumente, je treba dodatne argumente posredovati prek sklada. V zbirniku za procesor ARM Cortex-M4 s sintakso GNU v datoteki main.s napišite preprost podprogram sum6, ki vzame šest 32-bitnih celih števil in izračuna njihovo vsoto. Kot primer vzemite, da morate sešteti prvih šest naravnih števil: 1, 2, 3, 4, 5, 6 [5].

Rešitev:

Program v zbirniku (main.s)

```
.cpu cortex-m4
.syntax unified

        .section .text
        .global main

main:   mov  r0, #5         // 5. argument
        mov  r1, #6         // 6. argument
        mov  r2, #3         // 3. argument
        mov  r3, #4         // 4. argument
        push {r0, r1}       // potisni 5. in 6. argument na sklad
        mov  r0, #1         // 1. argument
        mov  r1, #2         // 2. argument
        bl   sum6
stop:   b    stop

sum6:   add  r0, r0, r1     // seštej 1. in 2. argument
        add  r0, r0, r2     // prištej 3. argument
        add  r0, r0, r3     // prištej 4. argument
        ldrd r2, r3, [sp]   // dvigni s sklada 5. in 6. argument
        add  r0, r0, r2     // prištej 5. argument
        add  r0, r0, r3     // prištej 6. argument
        bx   lr             // povratek v main

        .end
```

Program main postavi prve štiri argumente v registre r0, r1, r2 in r3 in z ukazom push {r0, r1} potisne zadnja dva argumenta na sklad. Pri tem se ne glede na vrstni red navedenih registrov v ukazu na sklad najprej potisne register z najvišjim indeksom, v našem primeru je to register r1.

Podprogram sum6 najprej akumulira vsoto prvih štirih argumentov v registru r0. Nato uporabi ukaz ldrd r2, r3, [sp] za naložitev dveh zaporednih 32-bitnih besed z vrha sklada v registra r2 in r3. Pri nalaganju vrednosti več registrov s sklada se najprej naloži vrednost v register z najnižjim indeksom, v našem primeru se torej vrednost z vrha sklada vnese v r2, vrednost pod njo pa v register r3. Ukaz ldrd, s katerim z vrha sklada dvignemo dve zaporedni 32-bitni besedi, bi lahko nadomestili z ukazoma ldr r2, [sp] in ldr r3, [sp, #4] za dvig samo po ene 32-bitne besede naenkrat. Beseda na vrhu sklada se začne na naslovu [sp], beseda tik pod vrhom sklada pa na naslovu [sp+4], saj so besede dolge 32 bitov ali 4 bajte. Podprogram sum6 vrne končni rezultat 21 (vsoto vseh šestih argumentov) v registru r0.

10.16 Programiranje v zbirniku za procesorje ARM (5/8)

Za razliko od naloge **10.14** morate zdaj napisati program za izračun dolžine istega niza Lep pozdrav!\n tako, da v datoteki main.c definirate niz in nato kličete podprogram v zbirniku za procesor ARM Cortex-M4 s sintakso GNU v datoteki strlen.s, ki izračuna dolžino niza. Znaka \null na koncu niza ne prištevajte k dolžini. Upoštevati morate protokol za klic zbirniškega podprograma, ki je definiran v binarnem vmesniku za vgrajene aplikacije ARM (EABI). Klicani zbirniški program pričakuje, da se mu argument (naslov niza) preda v registru r0. Klicatelj (program v C-ju) pričakuje, da podprogram v zbirniku vrne 32-bitni rezultat (dolžino niza) v register r0 [5].

Rešitev:

Program v jeziku C (main.c)

```
char str[25] =
          "Lep pozdrav!\n";

extern int strlen(char* s);

int main(void){
      int i;
      i = strlen(str);
      while(1);
}
```

Program v zbirniku (strlen.s)

```
.cpu cortex-m4
.syntax unified

    .section .text
    .global strlen

strlen:
    push {r4, lr}  // shrani r4 in lr
    mov r4, #0     // inicializacija dolžine

loop:
    ldrb r1, [r0, r4] // r0 = naslov niza
    cbz r1, exit // vejanje, če je znak \null
    add r4, r4, #1 // dolžina++
    b loop         // ponovitev zanke

exit:
    mov r0, r4     // postavi rezultat v r0
    pop {r4, pc}   // vrnitev v C

    .end
```

Izračunano dolžino niza (13) klicani zbirniški program vrne klicatelju (C-ju) v register r0. To vrednost dobi celoštevilčna spremenljivka i, program pa vstopi v neskončno zanko (ukaz while(1)).

10.17 Programiranje v zbirniku za procesorje ARM (6/8)

V nalogi **10.16** je C-jevski program klical podprogram za izračun dolžine niza v zbirniškem programu. Zdaj izračunajte dolžino pozdravnega niza v obratni smeri. V zbirniku za procesor ARM Cortex-M4 s sintakso GNU v datoteki main.s definirajte niz in pokličite C-jevsko funkcijo strlen v datoteki strlen.c. Zbirniški program mora slediti protokolu za klic procedure, definiranemu v binarnem vmesniku za vgrajene aplikacije ARM (EABI). Vhodni argument C-jevske funkcije (začetni naslov niza) mora postaviti v register r0 pred klicem funkcije. C-jevska funkcija mora zbirniškemu programu vrniti dolžino niza v registru r0. Znaka \null na koncu niza ne prištevajte k dolžini [5].

Rešitev:

Program v zbirniku (main.s)

```
.cpu cortex-m4
.syntax unified

    .section .text
    .global main
    .extern strlen

main:
    ldr r0, =niz
    bl  strlen      // klic funkcije strlen v C-ju
stop:
    b   stop        // neskončna zanka

    .section .data
niz:
    .asciz "Lep pozdrav!\n" // niz zaključen z
                            // null

    .end
```

Program v jeziku C (strlen.c)

```
int strlen(char *s){
  int i = 0;

  while (s[i] != '\0')
        i++;

  return i;
}
```

Pri tej nalogi je klicatelj zbirniški program, ki kliče C-jevsko funkcijo strlen. V C-ju izračunana dolžina niza se postavi v register r0 in program v zbirniku vstopi v neskončno zanko.

10.18 Programiranje v zbirniku za procesorje ARM (7/8)

Potrebujemo program za izračun vsote kvadratov prvih n naravnih števil.

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + n^2$$

Program naj bo realiziran v dveh ločenih datotekah, v C-jevski datoteki main.c in v zbirniški datoteki squareSum.s za procesor ARM Cortex-M4 s sintakso GNU. V datoteki main.c spremenljivki n dodelite vrednost in pokličite zbirniški podprogram squareSum, ki izračuna rezultat in ga nato vrne v C-jevski program. Podprogram v zbirniku mora slediti protokolu za klic procedure, definiranemu v binarnem vmesniku za vgrajene aplikacije ARM (EABI).

Rešitev:

Program v jeziku C (main.c)

```
int n = 5;

extern int squareSum(int n);

int main(void){
        int result;
        result = squareSum(n);

        while(1);

}
```

Program v zbirniku (squareSum.s)

```
.cpu cortex-m4
.syntax unified

        .section .text
        .global squareSum

squareSum:
        movs r1, #0         // vsota
        movs r2, #0         // števnik obhodov
zanke

loop:
        cmp r2, r0          // r0 = n
        bgt exit            // vejanje, če r2 > n
        mla r1, r2, r2, r1  // r1 = r2 * r2 + r1
        adds r2, r2, #1     // r2 = r2 + 1
        b loop              // ponovitev zanke

exit:
        movs r0, r1         // končni rezultat v r0
        bx lr               // vrnitev v C

        .end
```

Vhodni argument (n=5) za zbirniški podprogram squareSum se prenese prek registra r0. Tudi zbirniški program izračunani rezultat $\sum_{i=1}^{n} i^2 = 55$ vrne C-ju prek registra r0, ki se vpiše v spremenljivko result.

10.19 Programiranje v zbirniku za procesorje ARM (8/8)

Potrebujemo program, ki za nenegativno celo število n izračuna $n!$.

$$f(n) = \prod_{i=1}^{n} i = n \cdot (n-1) \cdot (n-2) \cdot \ldots \cdot 3 \cdot 2 \cdot 1$$

Program naj bo realiziran v dveh ločenih datotekah, v C-jevski datoteki main.c in v zbirniški datoteki factorial.s za procesor ARM Cortex-M4 s sintakso GNU. V datoteki main.c spremenljivki n dodelite vrednost in pokličite zbirniški podprogram factorial, ki izračuna $n!$ in nato rezultat vrne v C-jevski program. Podprogram v zbirniku mora slediti protokolu za klic procedure, definiranemu v binarnem vmesniku za vgrajene aplikacije ARM (EABI).

Rešitev:

Program v jeziku C (main.c)

```
int n = 5;

extern int factorial(int n);

int main(void){
       int result;
       result = factorial(n);

      while(1);
}
```

Program v zbirniku (factorial.s)

```
.cpu cortex-m4
.syntax unified

       .section .text
       .global factorial

factorial:
       movs r2, #1      // rezultat
       movs r1, #1      // i

       b check

loop:
       muls r2, r1, r2 // rezultat = i * rezultat
       adds r1, r1, #1 // i = i + 1

check:
       cmp r1, r0       // i = n?
       ble loop         // ne, ponovitev zanke

       movs r0, r2      // končni rezultat v r0
       bx lr            // vrnitev v C

       .end
```

Vhodni argument (n=5) za zbirniški podprogram factorial se prenese prek registra r0. Tudi zbirniški program izračunani rezultat 5! = 120 vrne C-ju prek registra r0, ki se vpiše v spremenljivko result.

A Priloga

A.1 Dokaz izreka o zvezi $R_A \geq R_G \geq R_H$ med aritmetično, geometrijsko in harmonično srednjo vrednostjo

Izrek. *Za $x_1, \ldots, x_n > 0$ velja*

$$R_A \geq R_G \geq R_H,$$

kjer

$$R_A := \frac{1}{n}\sum_{i=1}^{n} x_i, \quad R_G := \left(\prod_{i=1}^{n} x_i\right)^{1/n}, \quad R_H := \frac{n}{\sum_{i=1}^{n} \frac{1}{x_i}}.$$

Dokaz. *Funkcija $f(x) = \ln x$ je strogo konkavna na $(0, \infty)$, saj je*

$$f''(x) = -\frac{1}{x^2} < 0.$$

Jensenova neenakost za konkavno funkcijo f in enake uteži $w_i = \frac{1}{n}$ pravi:

$$f\left(\sum_{i=1}^{n} w_i x_i\right) \geq \sum_{i=1}^{n} w_i f(x_i).$$

***(1) Dokaz** $R_A \geq R_G$. Z uporabo $f = \ln$ in $w_i = \frac{1}{n}$ dobimo*

$$\ln\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \geq \frac{1}{n}\sum_{i=1}^{n} \ln x_i.$$

Ker je $R_A = \frac{1}{n}\sum_{i=1}^{n} x_i$, lahko naprej pišemo

$$\begin{aligned}
\ln R_A &\geq \frac{1}{n}\sum_{i=1}^{n} \ln x_i \\
&= \frac{1}{n}\ln\left(\prod_{i=1}^{n} x_i\right) \qquad \left(ker \sum_{i=1}^{n} \ln x_i = \ln\prod_{i=1}^{n} x_i\right) \\
&= \ln\left(\prod_{i=1}^{n} x_i\right)^{1/n} \\
&= \ln R_G.
\end{aligned}$$

Ker je funkcija e^x strogo naraščajoča, lahko obe strani neenačbe potenciramo z osnovo e in neenačaj se ohrani, sledi $R_A \geq R_G$.

***(2) Dokaz** $R_G \geq R_H$. Uporabimo Jensenovo neenakost na zaporedju $y_i := \frac{1}{x_i} > 0$:*

$$\ln\left(\frac{1}{n}\sum_{i=1}^{n} \frac{1}{x_i}\right) \geq \frac{1}{n}\sum_{i=1}^{n} \ln\left(\frac{1}{x_i}\right) = -\frac{1}{n}\sum_{i=1}^{n} \ln x_i = -\ln R_G = \ln\frac{1}{R_G}.$$

Po potenciranju obeh strani neenačbe z osnovo e dobimo

$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{x_i} \geq \frac{1}{R_G} \quad \Longrightarrow \quad R_G \geq \frac{n}{\sum_{i=1}^{n}\frac{1}{x_i}} = R_H.$$

S tem smo dokazali $R_A \geq R_G \geq R_H$. *Enakost nastopi v obeh neenakostih natanko tedaj, ko je* $x_1 = x_2 = \cdots = x_n$. □

A.2 Dokaz izreka $R_A \cdot R_H = R_G^2$ za srednje vrednosti nad samo dvema spremenljivkama

Izrek. *Naj bosta $x_1 > 0$ in $x_2 > 0$. Označimo*

$$R_A = \frac{x_1 + x_2}{2}, \quad R_H = \frac{2}{\frac{1}{x_1} + \frac{1}{x_2}}, \quad R_G = \sqrt{x_1 x_2}.$$

Tedaj velja identiteta

$$R_A \cdot R_H = R_G^2.$$

Dokaz. *Najprej poenostavimo R_H:*

$$R_H = \frac{2}{\frac{1}{x_1} + \frac{1}{x_2}} = \frac{2}{\frac{x_1 + x_2}{x_1 x_2}} = \frac{2x_1 x_2}{x_1 + x_2}.$$

Dobimo:

$$\begin{aligned} R_A \cdot R_H &= \frac{x_1 + x_2}{2} \cdot \frac{2x_1 x_2}{x_1 + x_2} \\ &= \frac{\cancel{x_1 + x_2}}{2} \cdot \frac{2x_1 x_2}{\cancel{x_1 + x_2}} \\ &= x_1 x_2 \\ &= \left(\sqrt{x_1 x_2}\right)^2 = R_G^2 \end{aligned}$$

□

A.3 Dokaz izreka o vsoti prvih n naravnih števil

Izrek. *Vsota prvih n naravnih števil je*

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \ .$$

Dokaz.
Izpeljava temelji na izračunu razlike kvadratov dveh sosednjih naravnih števil.

$$\sum_{i=1}^{n} i = 1 + 2 + 3 + \ldots + n = \ ?$$

$$(i-1)^2 = i^2 - 2i + 1$$
$$i^2 - (i-1)^2 = 2i - 1$$

Na levi in na desni strani enačbe izvedemo seštevanje za vse vrednosti i od 1 do n.

$$\sum_{i=1}^{n}(i^2 - (i-1)^2) = \sum_{i=1}^{n}(2i-1)$$

Izračunamo izraz na levi strani te enačbe.

$$\sum_{i=1}^{n}(i^2 - (i-1)^2) = n^2 - (n-1)^2 + (n-1)^2 - (n-2)^2 + (n-2)^2 - (n-3)^2 + \ldots$$
$$+ 3^2 - 2^2 + 2^2 - 1^2 + 1^2 - 0^2 = n^2$$

Dobimo:

$$n^2 = \sum_{i=1}^{n}(2i-1) = 2\sum_{i=1}^{n} i - \sum_{i=1}^{n} 1 = 2\sum_{i=1}^{n} i - n$$
$$2\sum_{i=1}^{n} i = n^2 + n = n(n+1)$$
$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

□

A.4 Dokaz izreka o vsoti kvadratov prvih n naravnih števil

Izrek. *Vsota kvadratov prvih n naravnih števil je*

$$\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6} \ .$$

Dokaz. *Izpeljava temelji na izračunu razlike kubov dveh sosednjih naravnih števil.*

$$\sum_{i=1}^{n} i^2 = 1^2 + 2^2 + 3^2 + \ldots + n^2 = ?$$
$$(i-1)^3 = i^3 - 3i^2 + 3i - 1$$
$$i^3 - (i-1)^3 = 3i^2 - 3i + 1$$

Na levi in na desni strani enačbe izvedemo seštevanje za vse vrednosti i od 1 do n.

$$\sum_{i=1}^{n}(i^3 - (i-1)^3) = \sum_{i=1}^{n}(3i^2 - 3i + 1)$$

Podobno kot pri vsoti razlik kvadratov sosednjih števil i in $i-1$ za prvih n naravnih števil (dokaz izreka v prilogi **A.3***) se vsi členi, razen prvega člena n^3, izničijo. Tako dobimo:*

$$n^3 = \sum_{i=1}^{n}(3i^2 - 3i + 1) = 3\sum_{i=1}^{n} i^2 - 3\sum_{i=1}^{n} i + \sum_{i=1}^{n} 1 = 3\sum_{i=1}^{n} i^2 - 3\frac{n(n+1)}{2} + n$$
$$3\sum_{i=1}^{n} i^2 = n^3 + 3\frac{n(n+1)}{2} - n$$
$$\sum_{i=1}^{n} i^2 = \frac{n^3}{3} + \frac{n(n+1)}{2} - \frac{n}{3} = \frac{2n^3 + 3n(n+1) - 2n}{6} = \frac{n(2n^2+3n+1)}{6}$$
$$= \frac{n(n+1)(2n+1)}{6}$$

□

Viri

[1] William Stallings, *Computer Organization and Architecture: Designing for Performance*, Ninth Edition, Pearson, 2013.

[2] John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, Fifth Edition, Morgan Kaufmann, 2012.

[3] Dimitrios A. Protopapas, *Microcomputer Hardware Design*, Prentice-Hall International, Inc., 1988.

[4] Barry B. Brey, *The Intel Microprocessors: 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium, and Pentium Pro Processor Architecture, Programming, and Interfacing*, Fifth Edition, Prentice-Hall, 2000.

[5] Yifeng Zhu, *Embedded Systems with ARM Cortex-M Microcontrollers in Assembly Language and C*, Second Edition, E-Man Press LCC, 2015.

Mikroračunalniške arhitekture: Zbirka rešenih nalog

Zmago Brezočnik
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, Maribor, Slovenija
zmago.brezocnik@um.si

Ta zbirka rešenih nalog je namenjena predvsem študentom kot pripomoček pri pripravi na kolokvije in pisne izpite z računskimi nalogami s področja mikroračunalniških struktur, mikroprocesorjev in mikrokrmilnikov. Naloge so razvrščene po poglavjih, ki obravnavajo zmogljivost računalnikov, komunikacijo med glavnimi komponentami računalnika, predpomnilnik, notranji in zunanji pomnilnik, vhodno-izhodno organizacijo mikroprocesorjev, nabore ukazov centralne procesne enote, načine naslavljanja in formate ukazov ter razvoj programov v zbirnem jeziku. Zadnje poglavje preverja znanje pri analizi odsekov kode in pisanju kratkih programov za mikroprocesorje iz družin Intel x86 in ARM.

DOI
https://doi.org/10.18690/um.feri.10.2025

ISBN
978-961-299-054-1

Ključne besede:
mikroračunalniške arhitekture, centralna procesna enota, pomnilnik, vhodno-izhodna organizacija mikroprocesorjev, zbirni jezik

Univerzitetna založba Univerze v Mariboru

DOI
https://doi.org/
10.18690/um.feri.10.2025

ISBN
978-961-299-054-1

Microcomputer Architectures: Collection of Solved Problems

Zmago Brezočnik
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia
zmago.brezocnik@um.si

Keywords:
microcomputer architectures, central processing unit, memory, input-output organization of microprocessors, assembly language

This collection of solved problems is primarily intended as a study aid for students preparing for midterm tests and written exams with calculation-based tasks in the field of microcomputer structures, microprocessors, and microcontrollers. The problems are organized into chapters covering computer performance, communication between main computer components, cache memory, internal and external memory, input-output organization of microprocessors, instruction sets of the central processing unit, addressing modes and instruction formats, and assembly language programming. The final chapter focuses on testing knowledge through the analysis of code segments and the writing of short programs for Intel x86 and ARM microprocessors.

University of Maribor Press

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko