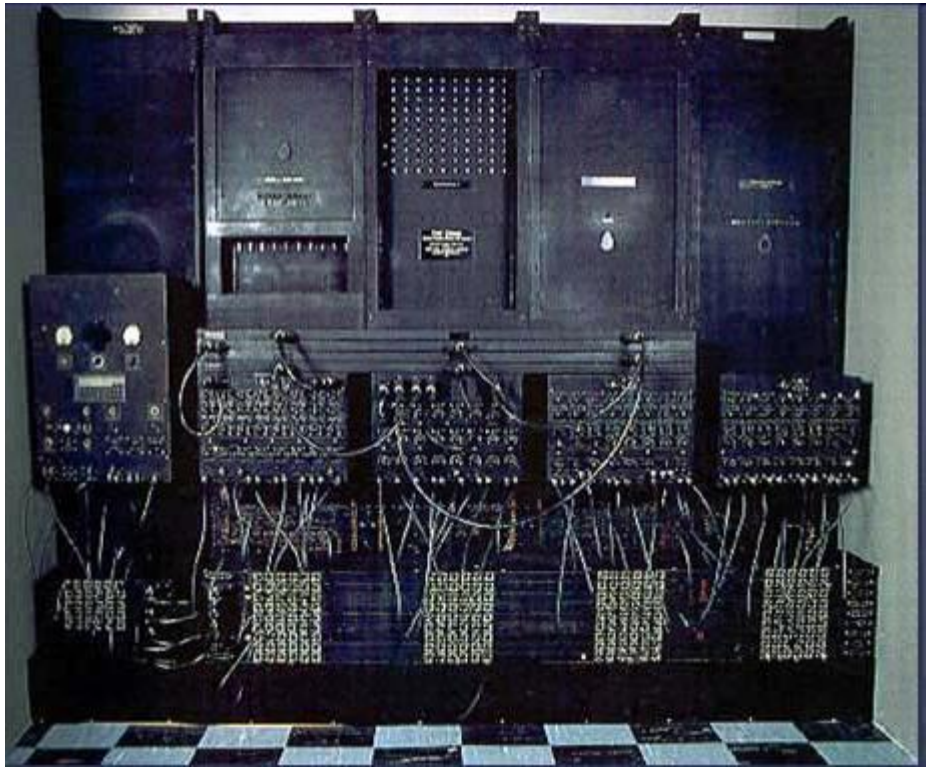


Princípy počítačov

Letný semester 2007/2008

Marián Novotný
(marian.novotny@upjs.sk)

http://s.ics.upjs.sk/~novmar/principy

http://s.ics.upjs.sk/~schmotze/Prp.html

Úrovne

- Súčasné počítače majú viac úrovní:
  1. Digital logic level (úroveň logických obvodov)
  2. Microprogramming level (mikroprogramová úroveň)
  3. Conventional machine level
  4. Operating system machine level (úroveň operačného systému)
  5. Assembly language level
     (úroveň jazyka symbolických adries, assemblera)
  6. Problem-oriented language level
     (úroveň vyšších programovacích jazykov)

Kódovanie

- Znakov
  - ASCII
  - Unicode
- Kompresné kódovanie
- Samoopravné kódy
- Čísel

Ascii kódovanie

- 7bitové, prvých 32 -riadiace znaky

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Unicode

ASCII/8859-1 Text

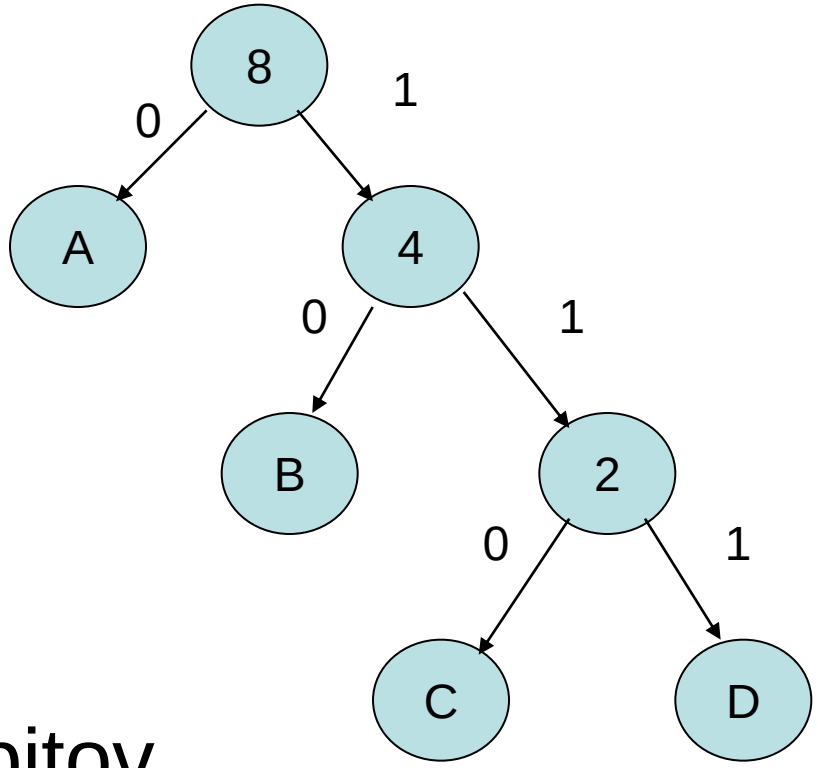
A	0100 0001
S	0101 0011
C	0100 0011
I	0100 1001
I	0100 1001
/	0010 1111
8	0011 1000
8	0011 1000
5	0011 0101
9	0011 1001
-	0010 1101
1	0011 0001
	0010 0000
t	0111 0100
e	0110 0101
x	0111 1000
t	0111 0100

Unicode Text

A	0000 0000 0100 0001
S	0000 0000 0101 0011
C	0000 0000 0100 0011
I	0000 0000 0100 1001
I	0000 0000 0100 1001
	0000 0000 0010 0000
天	0101 1001 0010 1001
地	0101 0111 0011 0000
	0000 0000 0010 0000
س	0000 0110 0011 0011
ل	0000 0110 0100 0100
ا	0000 0110 0011 0111
م	0000 0110 0100 0101
	0000 0000 0010 0000
α	0000 0011 1011 0001
κ	0010 0010 0111 0000
γ	0000 0011 1011 0011

Kompresné kódy

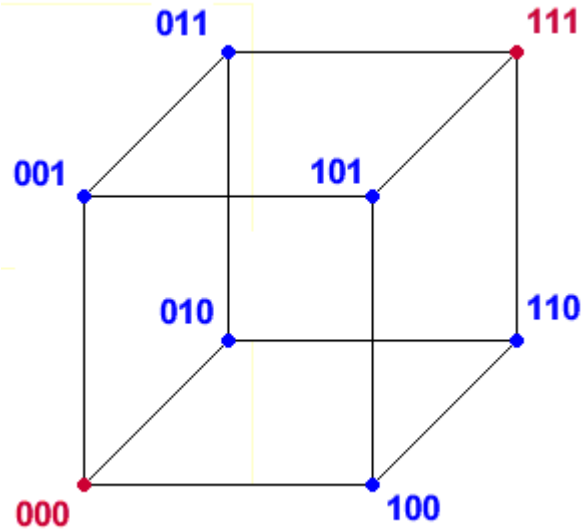
- BACADABA 8bytov
  - A 0
  - B 10
  - C 110
  - D 111



- 10011001110100 14bitov

Samoopravné kódy

- Detekcia chyby, oprava chyby



Pozičné sústavy čísel

- Každá číslica je charakterizovaná svojou polohou vzhľadom na rádovú čiarku, má rôznu váhu

$$r = a_n.Z^n + a_{n-1}.Z^{n-1} + \ldots + a_0.Z_0 + a^{-1}Z^{-1} + \ldots + a^{-m}Z^{-m}$$

Pr.

$$2008 = 2.10^3 + 0.10^2 + 0.10^1 + 8.10^0$$

$$3.14 = 3.10^1 + 1.10^{-1} + 4.10^{-2}$$

Nepozičné sústavy napr. rímska sústava MMVIII (1=I, 5=V, 10=X, 50=L, 100=C, 1000=M)

Prevody medzi sústavami

- 10-tková sústava: populárna u ľudí
- 2-ková: ľahko realizovateľná pomocou techniky
- 16-tková: (2^4=16), vhodnejšia pre ľudí
  - **Metóda postupného delenia**

$$r_C = c_m C^m + \ldots + c_1 C^1 + c_0$$

$$r_C = C.Q_1 + c_0$$

$$r_C = C.(c_m C^{m-1} + \ldots + c_1) + c_0$$

-**Metóda postupného odčítania**

-Od čísla r postupne odčítame násobky stále sa zmenšujúcich sa mocnín(najväčšie násobky mocnín, ktoré sú nanajvýš rovné prevádzanému číslu)

Metóda postupného odčítania

$$195_{10} = ?_2$$

$$2^7 = 128 \rightarrow 195 - 128 = 67 \mapsto 1$$

$$2^6 = 64 \rightarrow 67 - 64 = 3 \mapsto 1$$

$$2^5 = 32 > 3 \mapsto 0$$

$$2^4 = 16 > 3 \mapsto 0$$

$$2^3 = 8 > 3 \mapsto 0$$

$$2^2 = 4 > 3 \mapsto 0$$

$$2^1 = 2 \rightarrow 3 - 2 = 1 \mapsto 1$$

$$2^0 = 1 \rightarrow 1 - 1 = 0 \mapsto 1$$

$$195_{10} = 11000011_2$$

Metóda postupného delenia

$$195_{10} = ?_2$$

$$195 / 2 = 97 \rightarrow 195\%2 = 1 \mapsto 1$$

$$97 / 2 = 48 \rightarrow 97\%2 = 1 \rightarrow 1$$

$$48 / 2 = 24 \rightarrow 48\%2 = 0 \rightarrow 0$$

$$24 / 2 = 12 \rightarrow 24\%2 = 0 \rightarrow 0$$

$$12 / 2 = 6 \rightarrow 6\%2 = 0 \rightarrow 0$$

$$6 / 2 = 3 \rightarrow 6\%2 = 0 \rightarrow 0$$

$$3 / 2 = 1 \rightarrow 3\%2 = 1 \rightarrow 1$$

$$1 / 2 = 0 \rightarrow 1\%2 = 1 \rightarrow 1$$

$$195_{10} = 11000011_2$$

Prevod medzi sústavami so základom 16 a 2

- 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- 0,1

$$0_{16} = 0000_2$$
$$1_{16} = 0001_2$$
$$2_{16} = 0010_2$$
$$3_{16} = 0011_2$$
$$4_{16} = 0100_2$$
$$5_{16} = 0101_2$$
$$6_{16} = 0110_2$$
$$7_{16} = 0111_2$$

$$8_{16} = 1000_2$$
$$9_{16} = 1001_2$$
$$A_{16} = 1010_2$$
$$B_{16} = 1011_2$$
$$C_{16} = 1100_2$$
$$D_{16} = 1101_2$$
$$E_{16} = 1110_2$$
$$F_{16} = 1111_2$$

$$195_{10} = 1100 \mid 0011_2 = C3_{16}$$

Prevod desatinných čísel do dvojkovej sústavy

$$0.25_{10} = ?_2$$

$$0.25x2 = 0.5 < 1 \mapsto 0$$

$$0.5x2 = 1 - 1 = 0 \mapsto 1$$

$$0.25_{10} = 0.01_2$$

$$0.81_{10} = ?_2$$

$$0.81x2 = 1.62 - 1 = 0.62 \mapsto 1$$

$$0.62x2 = 1.24 - 1 = 0.24 \mapsto 1$$

$$0.24x2 = 0.48 \mapsto 0$$

$$0.48x2 = 0.96 \mapsto 0$$

$$0.96x2 = 1.92 - 1 = 0.92 \mapsto 1$$

$$0.92x2 = 1.84 - 1 = 0.84 \mapsto 1$$

Problémy

- Koľko cifier má prirodzené číslo x v desiatkovej sústave?
- Koľko cifier má prirodzené číslo x v dvojkovej sústave?
- Koľkokrát je dlhší zápis čísla v dvojkovej oproti zápisu v desiatkovej?
- Kedy má racionálne číslo kenečný neperiodický zápis v binárnej sústave?
- Ako previesť číslo zo 16-kovej sústavy do 4-kovej?

Obsah

- Kódovanie celých čísel
  - Priamy
  - Inverzný
  - Doplnkový
- Operácie
  - Sčítanie
  - Odčítanie
  - Pravidlá pretečenia
  - Rozšírenie dĺžky
- Kódovanie reálnych čísel a aritmetické operácie

Priamy kód

$$a_{n-1}a_{n-2}\ldots a_0$$

$a_{n-1} = 0$ kladné

$a_{n-1} = 1$ záporné

- 2 reprezentácie 0
- Počet hodnôt $2^N - 1$
- Rozsah $-(2^{N-1}-1)\ldots+2^{N-1}-1$

$$6_2 = 0110$$

$$-6_2 = 1110$$

Inverzný kód

- Záporné čísla

$-a = (2^N - 1) - a$ Negácia všetkých bitov(prečo?)

$a_{n-1}a_{n-2}\ldots a_0$

$a_{n-1} = 0$ kladné

$a_{n-1} = 1$ záporné

2 reprezentácie 0
Počet hodnôt $2^N - 1$
Rozsah $-(2^{N-1}-1)\ldots+ 2^{N-1}-1$

$$6_2 = 0110$$

$$-6_2 = 1001$$

Doplnkový kód

- Záporné čísla

$$-a = 2^N - a = (2^N - 1 - a) + 1$$

Negácia všetkých bitov, potom pripočítame 1

$a_{n-1}a_{n-2}\ldots a_0$

$a_{n-1} = 0$ kladné

$a_{n-1} = 1$ záporné

$$r = -a_{n-1}.2^{n-1} + \sum_{i=0}^{n-2}(a_i.2^i)$$

1 reprezentácia 0
Počet hodnôt 2^N
Rozsah $-2^{N-1}\ldots+2^{N-1}-1$

$$6_2 = 0110$$

$$-6_2 = 1010$$

Príklad kódovania(N=3)

Desiatková reprezentácia	priamy	doplnkový	inverzný
+3	011	011	011
+2	010	010	010
+1	001	001	001
0	000,100	000	000,111
-1	101	111	110
-2	110	110	101
-3	111	101	100
-4	-	100	-

Aritmetické operácie

Sčítanie, odčítanie, násobenie, delenie binárnych čísel podobné operáciám v desiatkovej sústave

Sčítanie $c_i = a_i + b_i + z_{i-1} \quad z_i \text{ prenos}, z_{-1} = 0$

- Priamy kód
  - Sčítanie, odčítanie komplikované, treba rozlíšiť prípady
  - Jednoduchá konverzia pre uloženie s m bitmi (m>n) (ako?)
- Doplnkový kód
  - Sčítanie funguje pre všetky čísla
  - Odčítanie X-Y=X+(-Y), nájdeme opačnú hodnotu Y (negáciou Y a pripočítaním 1) a sčítame s X

Doplnkový kód

- Predĺženie na m bitov m>n

16bitov	32bitov
0000 0100 1101 0100	0000 0000 0000 0000 0000 0100 1101 0100
1101 0101 0010 0011	1111 1111 1111 1111 1101 0101 0010 0011

- Pretečenie nastane vtedy a len vtedy, ak sčítanie čísel s rovnakým znamienkom dá výsledok s opačným znamienkom

Inverzný kód(N=4)

	1	0001
+	-2	1101
		1110

	2	0010
+	-2	1101
+	1	1111
		0000

	1	0001
+	2	0010
	3	0011

	-1	1110
+	-2	1101
+	1	1011
	-3	1100

	2	0010
+	-1	1110
+	1	0000
	1	0001

Reálne čísla

- Formát s pevnou rádovou čiarkou

$$r = a_n.Z^n + a_{n-1}.Z^{n-1} + \ldots + a_0.Z_0 + a^{-1}Z^{-1} + \ldots + a^{-m}Z^{-m}$$

  - Sčítanie, odčítanie ako pri celých, potrebné odseknutie, alebo zaokrúhlenie, ak prekročíme rozsah
- Formát s pohyblivou rádovou čiarkou
  - Analógia vedeckého zápisu čísel

$$r = M * z^E \quad M \text{ mantisa}, z \text{ základ}, E \text{ exponent}$$

Formát s pohyblivou rádovou čiarkou

+(-)	exponent	mantisa

- základ z
- Počet cifier na reprezentáciu mantisy m
- Spôsob kódovania mantisy (znamienka)
- Sústava pre kódovanie exponenta Se
- Počet cifier na reprezentáciu exponenta e
- $M_{min}(M_{max})$- minimálna(maximálna) mantisa
- $H_{min}(H_{max})$ –minimálna(maximálna) hodnota v danej reprezentácii
- Normalizovaná mantisa v tvare x.y , kde x je nenulové a y je zvyšná časť v binárnej sústave x=1 nemusíme písať – technika skrytého bitu

Príklad 32 bitov

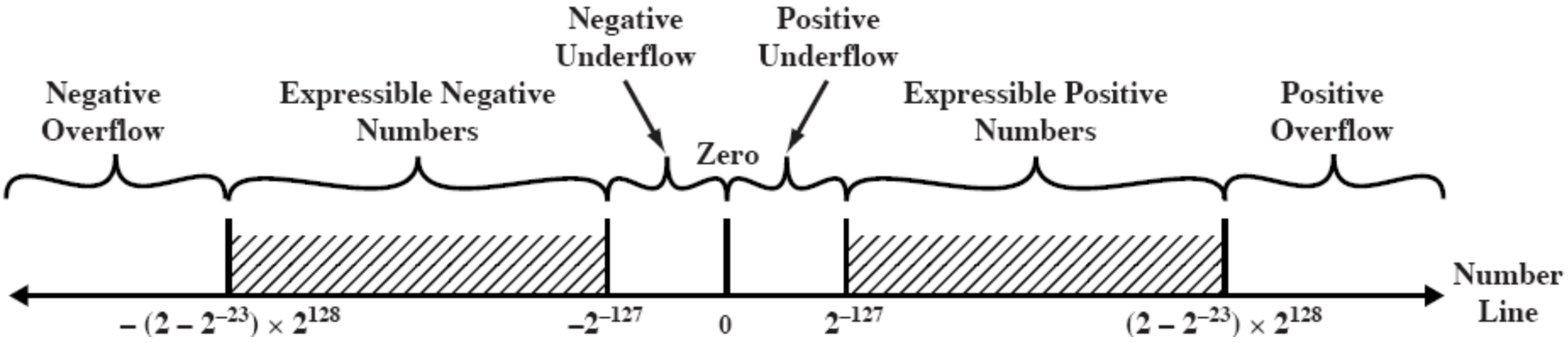
- z=2, Se=2, m=24(skrytý bit), e=8 – kódované binárne v rozsahu od -127…128

$$M_{\min} = 1.000..._2 = 1$$

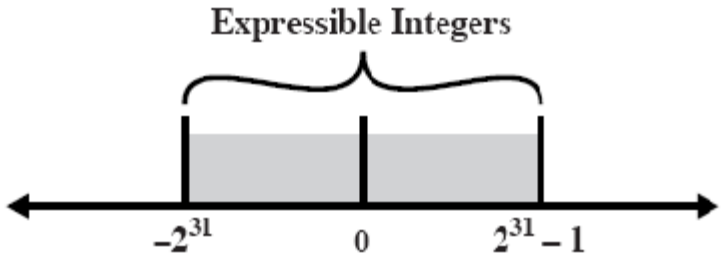
$$M_{\max} = 1.111..._2 = 2 - 2^{-23}$$

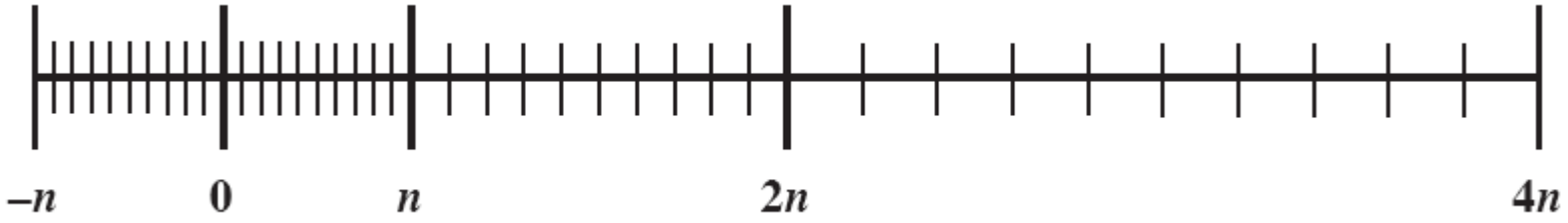
$$H_{\min} = (1) * 2^{-127}$$

$$H_{\max} = (2 - 2^{-23}) * 2^{+128}$$



Hustota čísel





Sčítanie

- A+B=?
- A(B) má mantisu $M_A(M_B)$, exponent $E_A(E_B)$
- $E_A=E_B$ – sčítame mantisy, upravíme výsledok do normalizovaného tvaru
- $E_A>E_B$

$$A+B=M_A*2^{E_A}+M_B*2^{E_B}=(M_A*2^{E_A-E_B}+M_B)*2^{E_B}$$

Potom znormalizujeme

Násobenie

- A*B=?
- A(B) má mantisu M_A(M_B), exponent E_A(E_B)

$$A*B = M_A * 2^{E_A} * M_B * 2^{E_B} = (M_A * M_B) * 2^{E_A + E_B}$$

Potom znormalizujeme

Delenie

- A+B=?
- A(B) má mantisu M_A(M_B), exponent E_A(E_B)

$$A/B = M_A * 2^{E_A} / (M_B * 2^{E_B}) = (M_A / M_B) * 2^{E_A - E_B}$$

Potom znormalizujeme

Obsah

- Ďalšie kódovania čísel
  - BCD kód
  - Grayov kód
- Úvod do logických funkcií
  - Základné pojmy,unárne a binárne funkcie
  - Základné vlastnosti
  - Normálne formy, vhodnosť použitia
    - DNF
    - CNF
  - Zjednodušovanie zápisu logickej funkcie
    - Algebraická minimalizácia
    - Karnaughova metóda

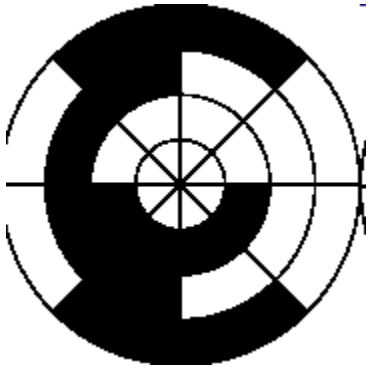
BCD kód

- Výhodný ak je potrebné často konvertovať čísla medzi desiatkovou a dvojkovou sústavou(napr. kalkulačky)
- Pomocou 4 bitov sa zakóduje 1 cifra v 10tkovej
- 0000=0,…,1001=9, zvyšok nevyužité
- Pomocou N bitov môžeme reprezentovať $\sim 10^{N/4}$ hodnôt

Sčítanie čísel v BCD kóde

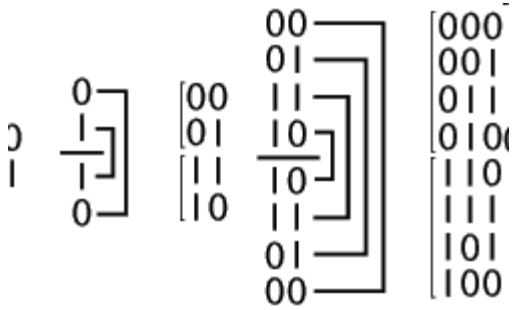
- Pomocou inštrukcií pre sčítanie bin.čísel bez znamienka sčítame čísla
- Prevedieme korekciu výsledku
  - Ak je v bloku číslo >9, k bloku pripočitáme 6(prečo?)
  - Ak najvyššie 4 bity majú hodnotu väčšiu ako 9, došlo k pretečeniu

Grayov kód



- Kód nasledovníka sa líši v jednom bite

Desiatkova	Dvojkova	Grayov kód
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100



Logické funkcie

- výrok – tvrdenie, ktoré môže byť pravdivé, nepravdivé
- Pomocou logických spojok vytvárame nové výroky
- Množinu {pravda,nepravda} môžeme reprezentovať napr. pomocou množiny {0,1}
- Funkciu $f:\{0,1\}^N \rightarrow \{0,1\}$ nazývame logickou funkciou N premenných

Unárne a binárne logické funkcie

X	y	NOT x	x AND y	x OR y	x XOR y	x NAND y	x NOR y
0	0	1	0	0	0	1	1
1	0	0	0	1	1	1	0
0	1	1	0	1	1	1	0
1	1	0	1	1	0	0	0

- Unárnych – $2.2=4$
- Binárnych – $2.2.2.2=16$
- N-árnych – 2^{2N} (prečo?)

Základné vlastnosti logických funkcií

- Dva výrazy nazývame ekvivalentnými(=), ak pre všetky ohodnotenia premenných nadobúdaju rovnakú hodnotu

x+0=x	x.0=0
x+1=1	x.1=x
x+x=x	x.x=x
x+x’=1	x.x’=0
(x+y)’=x’.y’	(x.y)’=x’+y’

x+y=y+x
x.y=y.x
x+y+z=(x+y)+z=x+(y+z)
x.y.z=(x.y).z=x.(y.z)
(x+y).z=x.z+yz
x+(y.z)=(x+y).(x+z)

Normálne formy

- Literálom nazývame premennú alebo negáciu premennej
- Výraz je v disjunktívnej normálnej forme(DNF), ak sa skladá zo súčtu podvýrazov, ktoré sú v tvare súčinu navzájom rôznych literálov
  - Napr. (xyz)+(x'yz)+(xy'z') je v DNF
- Výraz je v konjuktívnej normálnej forme(CNF), ak sa skladá zo súčinu podvýrazov, ktoré sú v tvare súčtu navzájom rôznych literálov
  - Napr. (x+y+z).(x'+y+z).(x+y'+z') je v CNF
- Každá logická funkcia sa dá vyjadriť v DNF(CNF)

Disjunktívna normálna forma

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

x	y	z	f(x,y,z)	výraz
0	1	0	1	x’.y.z’
1	0	0	1	x.y’.z’
1	1	0	1	x.y.z’
1	1	1	1	x.y.z

(x’.y.z’)+(x.y’.z’)+(x.y.z’)+(x.y.z)

Konjuktívna normálna forma

x	y	z	f(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

x	y	z	f(x,y,z)	výraz
0	0	0	0	x+y+z
0	0	1	0	x+y+z’
0	1	1	0	x+y’+z’
1	0	1	0	x’+y+z’

(x+y+z).(x+y+z’).(x+y’+z’).(x’+y+z’)

Vhodnosť reprezentácie

- Problém splniteľnostivosi, pravdivosti
- V DNF je ľahko rozhodnúť splniteľnosť a ťažko pravdivosť
- V CNF je ľahko rozhodnúť pravdivosť a ťažko splniteľnosť

Algebraická minimalizácia

- zakladá sa na algebraickej úprave výrazov, využívajúc vzťahy medzi logickými funkciami
- Nie je systematická, založená na hádaní vzťahu

$$f = \bar{x}yz + x\bar{y}z + xy\bar{z} + xyz$$

$$f = (\bar{x}yz + xyz) + (x\bar{y}z + xyz) + (xy\bar{z} + xyz) =$$

$$= yz(\bar{x} + x) + xz(\bar{y} + y) + xy(\bar{z} + z) =$$

$$= yz + xz + xy$$

Karnaughova metóda

- Využíva spájanie súčinov, ktoré sa líšia v jednej premennej
- xyzt+xyzt’=xyz(t+t’)=xyz
- Vytvoríme mapu, ktorá obsahuje 2^n políčok
- Susedia políčka sú políčka, ktoré sa líšia práve v jednej premennej
- Hľadáme v mape oblasti 2, 4, 8, 2^n susedných políčok jednotiek, aby sme zo skupín súčinov vylúčili 1, 2,3, rsp. n premenných

Karnaughova mapa

x	y	z	t	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

$zt \backslash xy$	00	01	11	10
00	1	0	0	1
01	0	1	1	1
11	0	0	0	1
10	1	0	0	1

x’yz’t+xyz’t=(x+x’)yz’t=yz’t

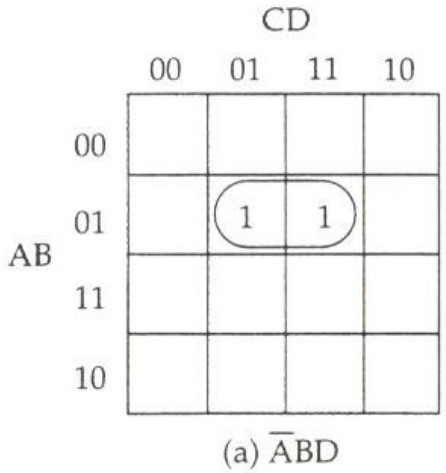
Susedia políčka

$zt \backslash xy$	00	01	10	11
00		X		
01	X	■	X	
10		X		
11				p

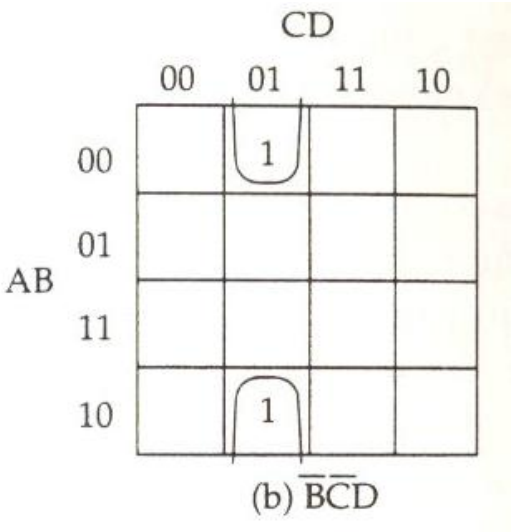
$zt \backslash xy$	00	01	10	11
00		X		
01				
10		X		
11	X	■	X	

$zt \backslash xy$	00	01	10	11
00	X			
01				
10	X			
11	■	X		X

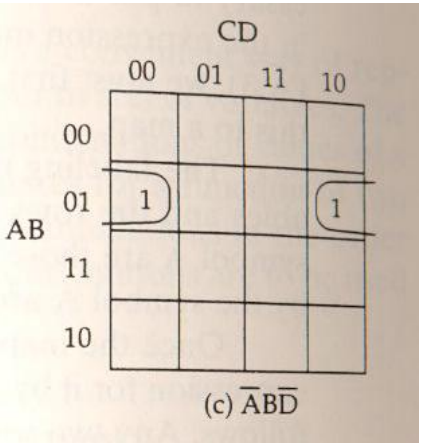
Dvojice v Karnaughovej mape



(a) $\overline{A}BD$

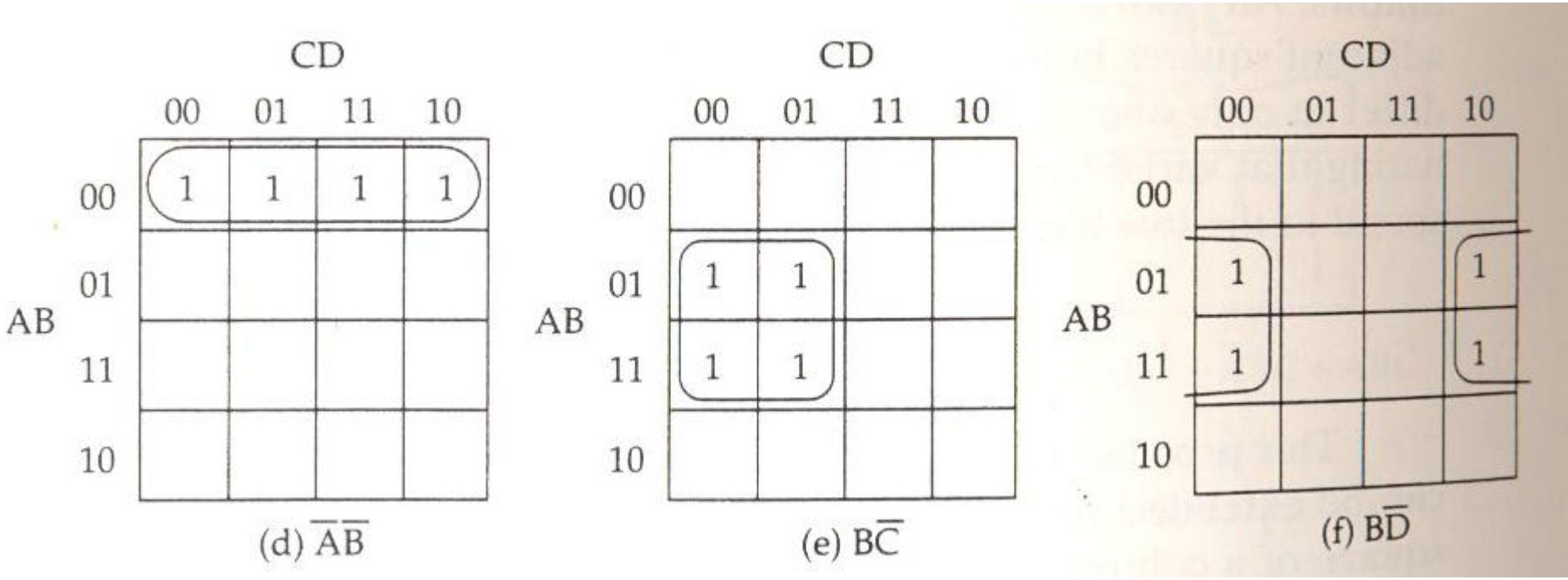


(b) $\overline{B}\overline{C}D$

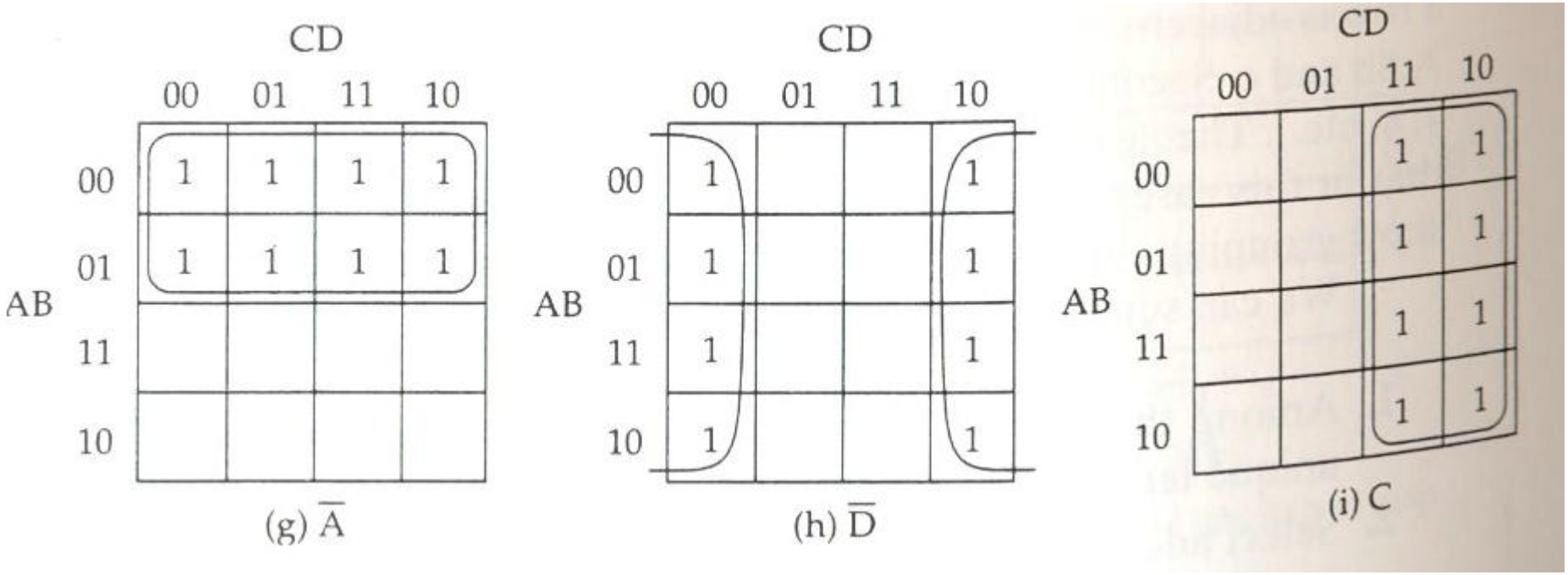


(c) $\overline{A}B\overline{D}$

Štvorice v Karnaughovej mape



8 políčok v Karnaughovej mape

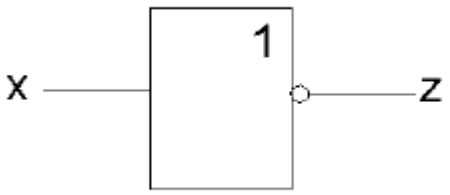


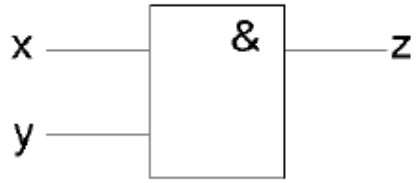
Obsah

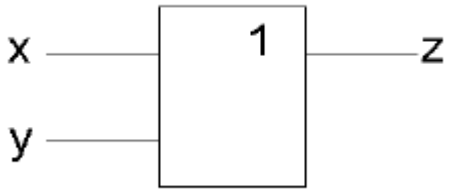
- Základné hradlá
- Syntéza logických obvodov
- Zjednotenie, prienik, doplnok
- Výhybka, testovanie parity
- Dekóder, prioritný kóder
- Multiplexor, demultiplexor
- Porovnávací obvod
- Sčítačka
- Aritmeticko-logická jednotka

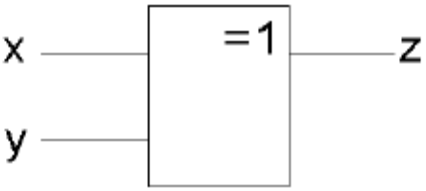
Kombinačné obvody

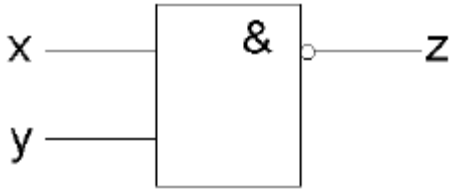
- Každý kombinačný obvod sa dá popísať logickou funkciou a pre každú log. funkciu existuje obvod
- Základné hradlá realizujú základné log.funkcie

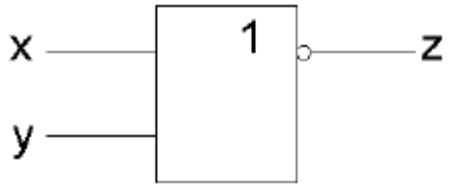












Návrh obvodov

- Efektívnosť obvodov
  - Priestorová (počet obvodov), časová zložitosť
- Napr. &(A,B,C,D,E,F,G,H)=
  - A & (B &(C &(D &(E&(F&(G&H))))))
  - ((A&B)&(C&D))&((E&F)&(G&H))
  - rozdielna hĺbka schémy(časová zložitosť)
- Syntéza
  - Navrhneme jednoduchšie a z nich sa budú skladať zložitejšie

Zjednotenie, prienik, doplnok

- Zjednotenie
  - C=A+B
  - Pomocou OR hradla
  - Spájanie informácií
- Prienik
  - C=A.B
  - Pomocou AND hradla
  - Rozdelenie informácie podľa tzv. masky
- Doplnok
  - C=A’
  - Pomocou NOT hradla
  - Pri doplnkovom kóde, maskovanie prerušení

Výhybka, testovanie parity

- Výhybka
  - $V=A.S_A + B.S_B$
  - Na riadenie výberu

- Testovanie parity
  - Pri odosielaní treba určiť hodnotu paritného bitu
  - Pri príjme skontrolovať správnosť dát
  - Využijeme vlastnosť funkcie XOR
    - =1, ak je počet jednotiek nepárny
    - =0, ak je počet jednotiek párny

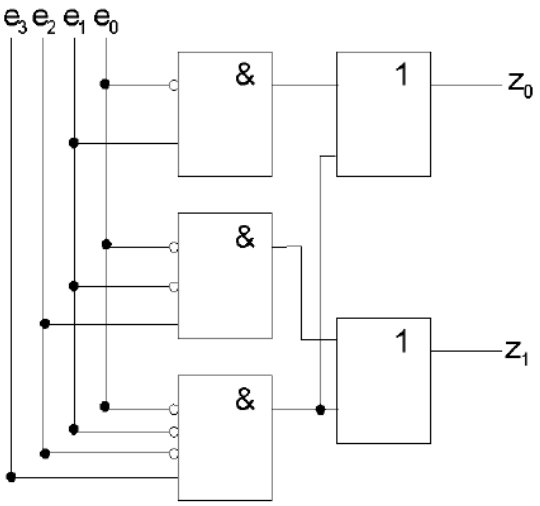
S_A	S_B	V
0	0	0
0	1	B
1	0	A
1	1	?

Dekóder, prioritný kóder

- Dekóder
  - Pri dekódovaní inštrukcií, pri zápise a čítaní do pamäte, zisťovanie adries

z_{n-1}	...	$z_1 z_0$	e_{2^n-1}	.....	$e_1 e_0$
0	...	0 0	0	.....	0 1
0	...	0 1	0	.....	1 0
	.			.	
	.			.	
	.			.	
1	...	1 1	1	.....	0 0

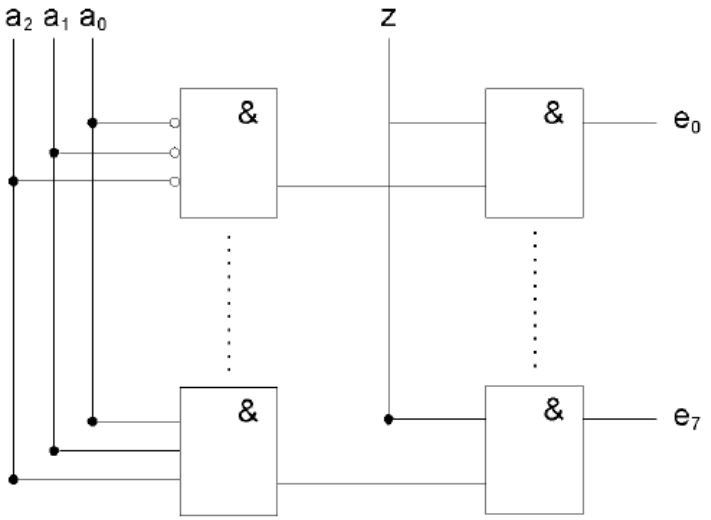
- Prioritný kóder



e_{2^n-1}	.....	$e_1 e_0$	z_{n-1}	...	$z_1 z_0$
0	.....	0 1	0	...	0 0
0	.....	1 0	0	...	0 1
	.			.	
	.			.	
	.			.	
1	.....	0 0	1	...	1 1

Multiplexor

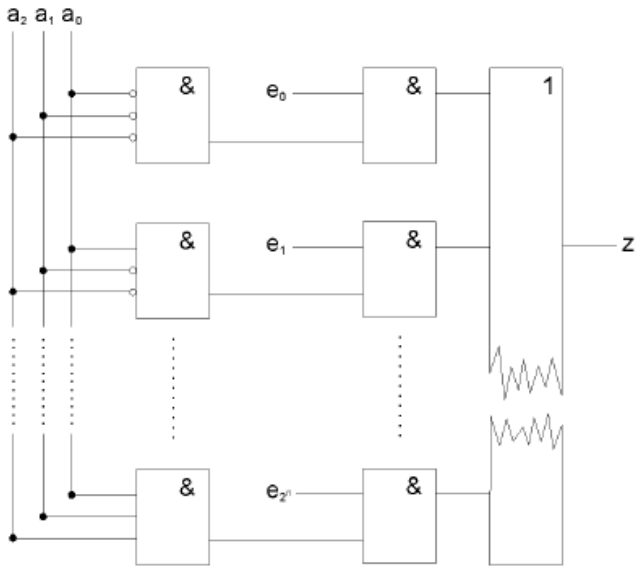
e_{2^n-1}	.....	$e_1 e_0$	a_{n-1}	...	$a_1 a_0$	z
–	.....	– s	0	...	0 0	s
–	.....	s –	0	...	0 1	s
	.			.		.
	.			.		.
	.			.		.
s	.....	– –	1	...	1 1	s



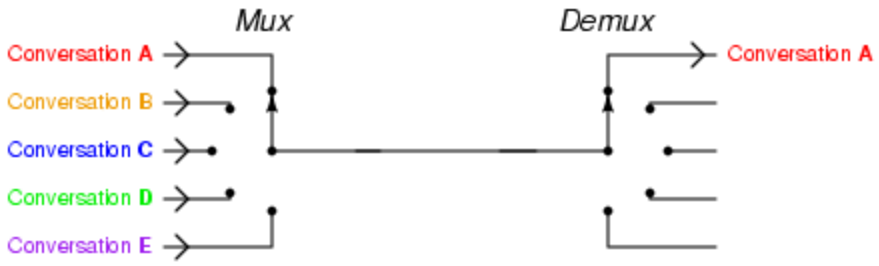
- Zovšeobecnená výhybka
- Prevod sériového na paralelný
- Prepojenie registrov
- Dá sa realizovať aj pomocou dekódera

Demultiplexor

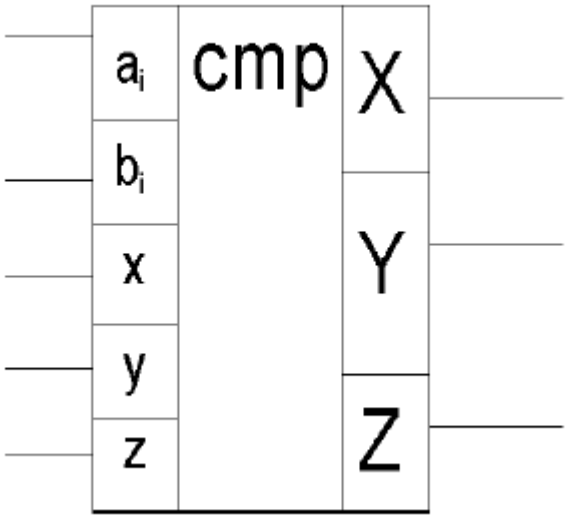
z	a_{n-1}	...	$a_1 a_0$	e_{2^n-1}		.....	$e_1 e_0$
s	0	...	0 0	0	0	.....	0 s
s	0	...	0 1	0	0	.....	s 0
.		.				.	
.		.				.	
.		.				.	
s	1	...	1 1	s	0	.....	0 0



- Spojenie multi a demultiplexora



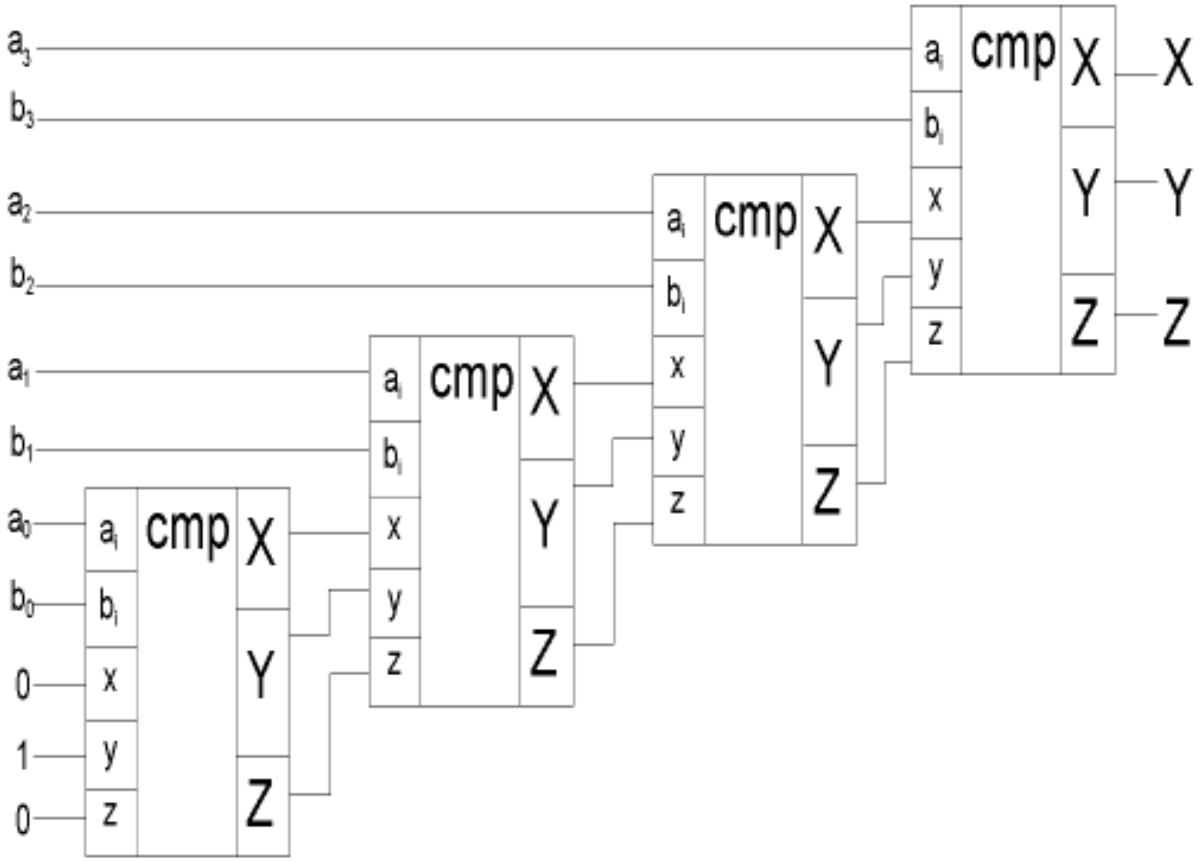
Jednobitový porovnávací obvod



a_i	b_i	x	y	z	X	Y	Z
0	0	1	0	0	1	0	0
0	0	0	1	0	0	1	0
0	0	0	0	1	0	0	1
0	1	–	–	–	1	0	0
1	0	–	–	–	0	0	1
1	1	1	0	0	1	0	0
1	1	0	1	0	0	1	0
1	1	0	0	1	0	0	1

X=1, ak B>A Z=1, ak A>B Y=1, ak A=B

Porovnávací obvod



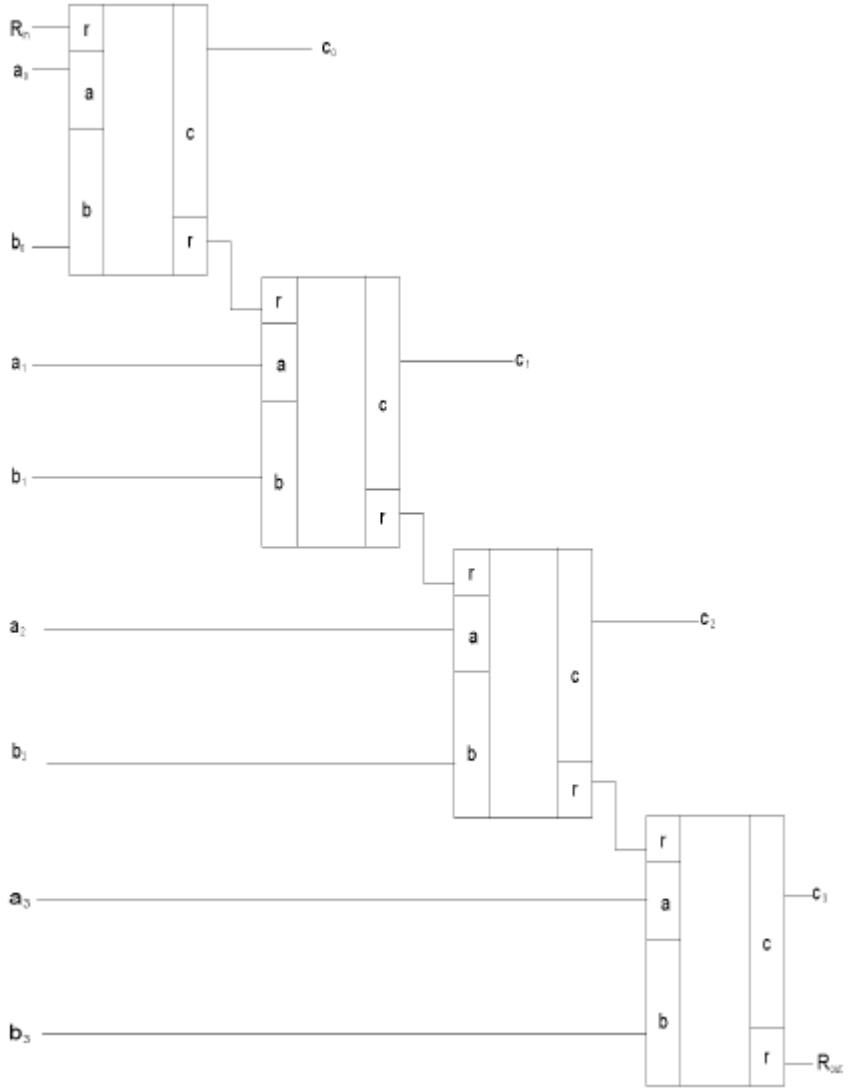
Jednobitová sčítačka

a_i	b_i	r_i	c_i	r_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$r_{i+1}=a_ib_i+a_ir_i+b_ir_i$

$c_i=a_i$ XOR b_i XOR r_i

Sčítačka so sériovým prenosom



Sčítačka so zrýchleným prenosom

- Vzťahy pre prenosy

$$r_0 = a_0 b_0$$
$$r_1 = a_1 b_1 + a_1 a_0 b_0 + b_1 a_0 b_0$$
$$r_2 = a_2 b_2 + a_2 r_1 + b_2 r_1$$
$$r_k = a_k b_k + r_{k-1} \cdot (a_k + b_k)$$

- Počet obvodov
  - R_0=1, R_1=7,
  - R_i=2.R_{i-1} +4 pre i>1
  - $R_i \sim 2^i$

Štruktúra	Priestorová zložitosť ∾	Časová zložitosť
1×32	2^{32}	1
2×16	$2 \cdot 2^{16}$	2
4×8	$4 \cdot 2^8$	4
8×4	$8 \cdot 2^4$	8
32×1	$32 .2$	32

Jednoduchá ALU

- Na vstupe sú vstupné dáta a riadiace signály (MS_0…S_3)
- Aritmetické

$$F = A + B \qquad F = A + B - 1 \qquad F = A$$
$$F = A - B \qquad F = A + 1 \qquad F = A + A$$
$$F = A + B + 1 \qquad F = A - 1 \qquad F = 1$$

- Logické

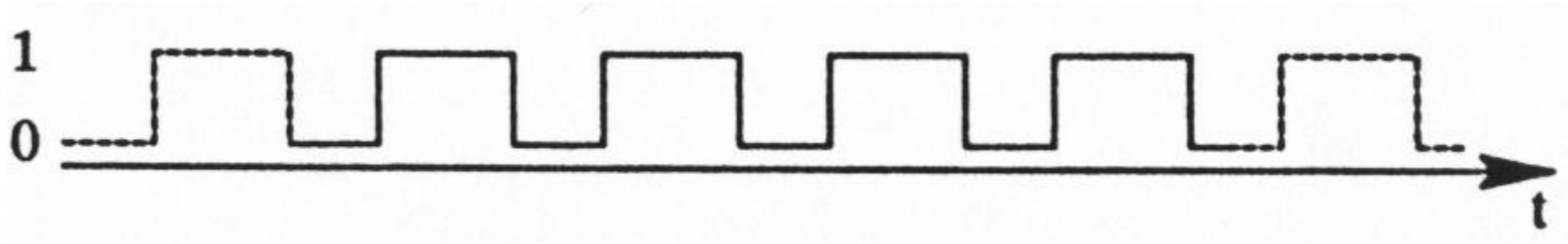
$$F = A' \qquad F = B' \qquad F = A \qquad F = (AB)'$$
$$F = AB \qquad F = 1 \qquad F = B \qquad F = A \oplus B$$
$$F = A + B \qquad F = 0 \qquad F = (A + B)' \qquad F = (A \oplus B)'$$

Obsah

- Základná charakteristika
- Konečnostavový automat
- Asynchrónne a synchrónne obvody
- Klopné obvody SR, D, JK
- Register
- Čítačka
- Pamäť

Sekvenčné obvody

- Sú závislé na predchádzajúcich vstupoch
- Obvod má určitý vnútorný stav, pracuje v krokoch(taktoch) t.j. v diskrétnom čase



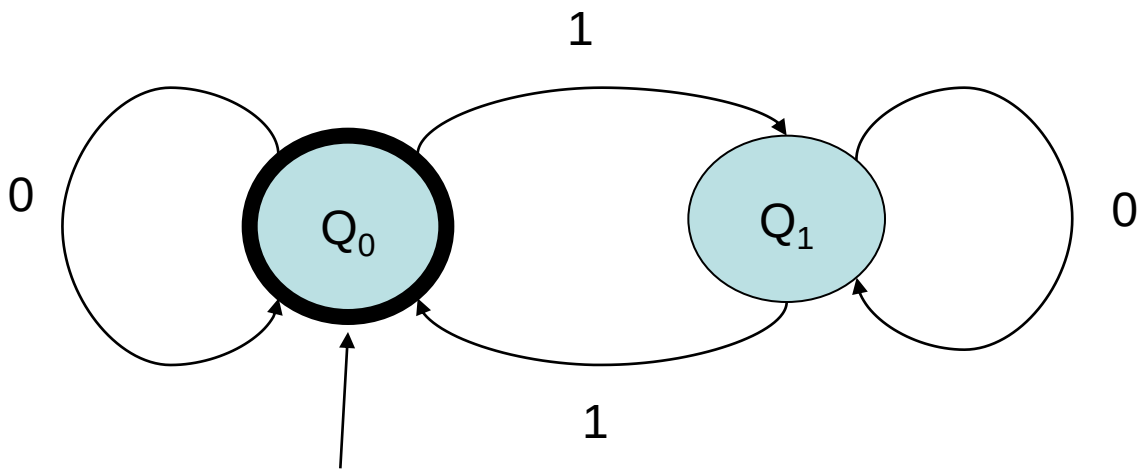
- Matematickým modelom je konečnostavový automat
- Synchrónne: má špeciálny krokovací vstup. Obvod prejde do nasledujúceho stavu až v prítomnosti krokového signálu. Pomalšie ako asynchrónne – taktovacia frekvencia sa prispôsobí najpomalšej časti
- Asynchrónne: nie sú časovo zladené so vstupom, môže dôjsť ku vytvoreniu nevhodného stavu

Konečnostavový automat

- Má konečnú pamäť - množinu stavov Q
- Pracuje v krokoch podľa programu, má na vstupe konečnú postupnosť znakov, začína nejakým stavom a má množinu stavov, v ktorej automat akceptuje vstup
- Formálne je to pätica (Q,E,D,q_0,F), kde Q je množina stavov, E vstupná abeceda, D prechodová funkcia (zobr. D:QxE->Q), q_0 počiatočný stav a F množina koncových stavov

Príklad automatu

- Stroj, ktorý akceptuje reťazec s párnym počtom jednotiek
- $E=\{0,1\}$, $Q=\{Q_0,Q_1\}$, $F=\{Q_0\}$, $q_0=Q_0$



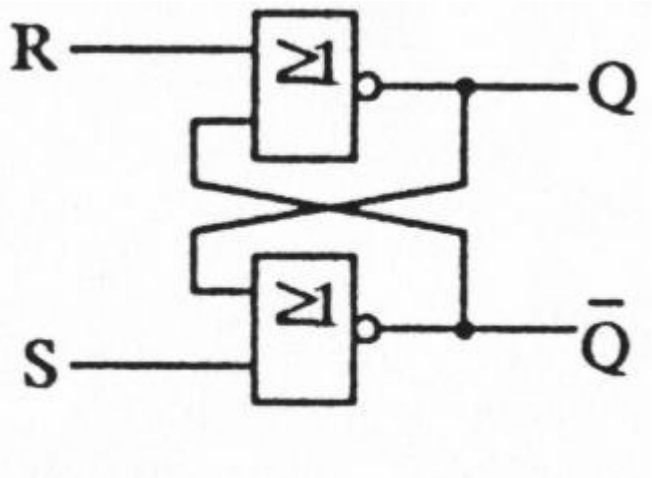
Moorov a Mealyho automat

- Máme navyše výstupnú abecedu G, výstupnú funkciu o
  - Moorov stroj
    - o: Q->G
  - Mealyho stroj
    - o:QxE->G

Pr.Máme automat na kávu a čokoládu. Káva stojí 10SK a čokoláda 15. Automat berie len desať a päťkorunáčky, je možná len jedna objednávka a v prípade potreby vydáva peniaze.
E={5,10,Káva, Čokoláda, Zruš} G={Vráť5,Vráť10, Vráť15, Vráť20, DajKávu, Dajčokoládu}

SR klopný obvod

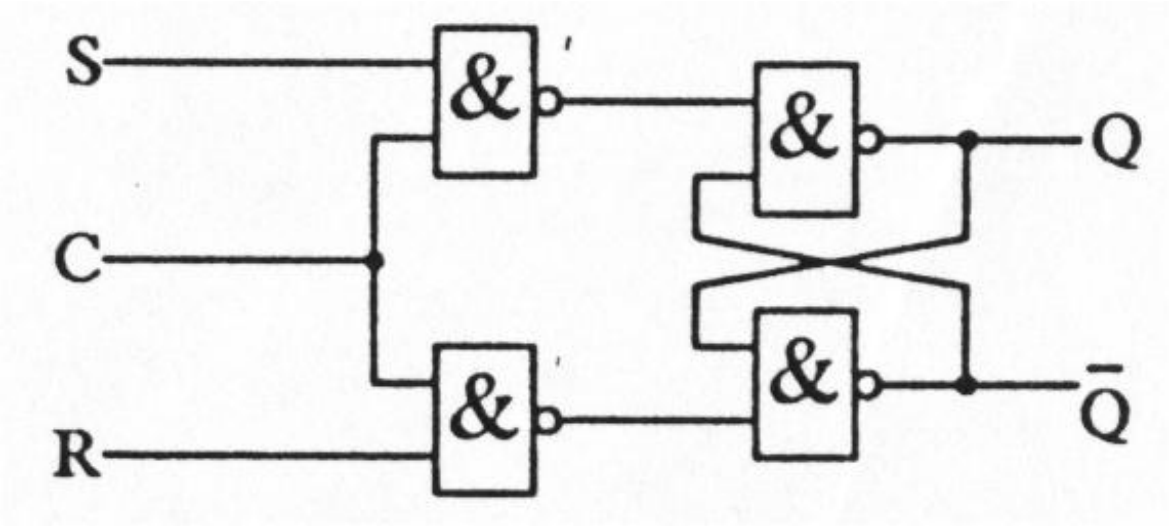
- Základná funkcia je pamätať si svoj stav
- Vieme zmeniť pomocou Set(1), Reset(0)
- Asynchrónny, chybné stavy, ak Q=Q’, nedefinovaný pre S=1, Q=1



S	R	Q	$\underline{Q}$	Stav
1	0	1	0	Set
0	0	1	0	Pamäťový stav
0	1	0	1	Reset
0	0	0	1	Pamäťový stav
1	1	0	0	Nedefinované

Synchrónny RS

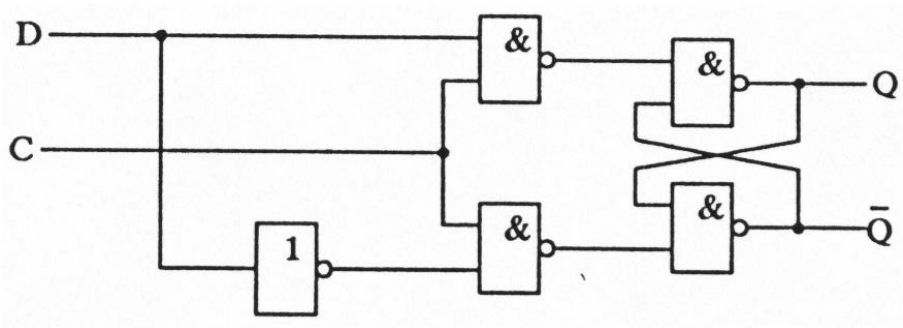
Kontrólny vstup C



C	S	R	Stav obvodu
0	x	x	Bezo zmien
1	0	0	Pamäťový stav
1	0	1	Q=0, stav Reset
1	1	0	Q=1, stav Set
1	1	1	Nedefinované

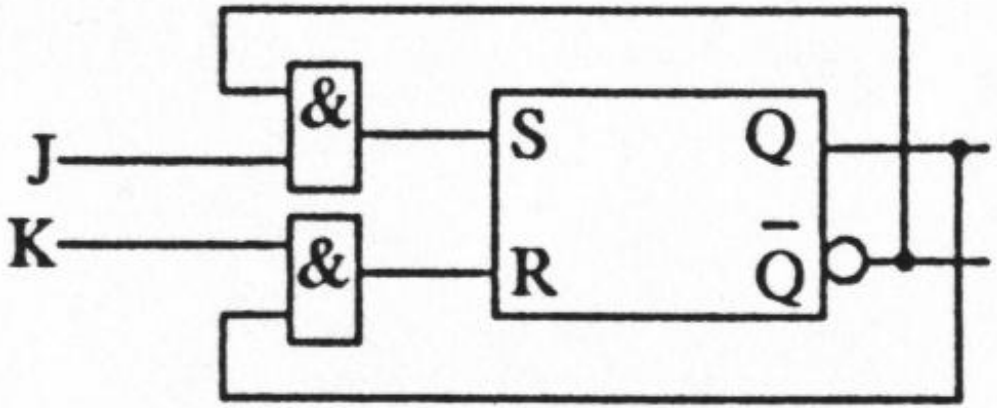
Klopný obvod D

- Aby sa zamedzilo nedefinovanému stavu



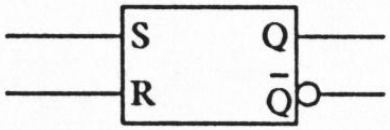
C	D	Ďalší stav
0	X	Bezo zmien
1	0	Q=0, Reset
1	1	Q=1,Set

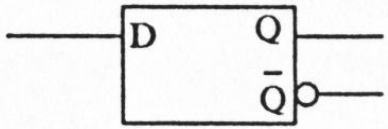
Klopný obvod JK

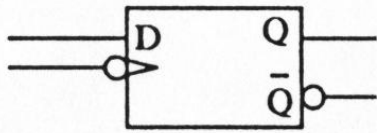


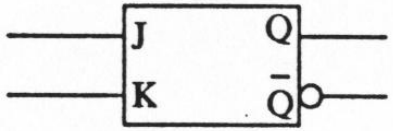
J	K	Q_{t-1}	Q_t
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Označenie

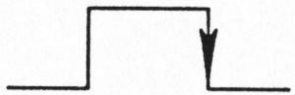




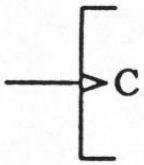




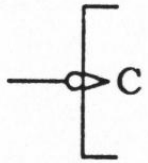
nábežná hrana



zostupná hrana



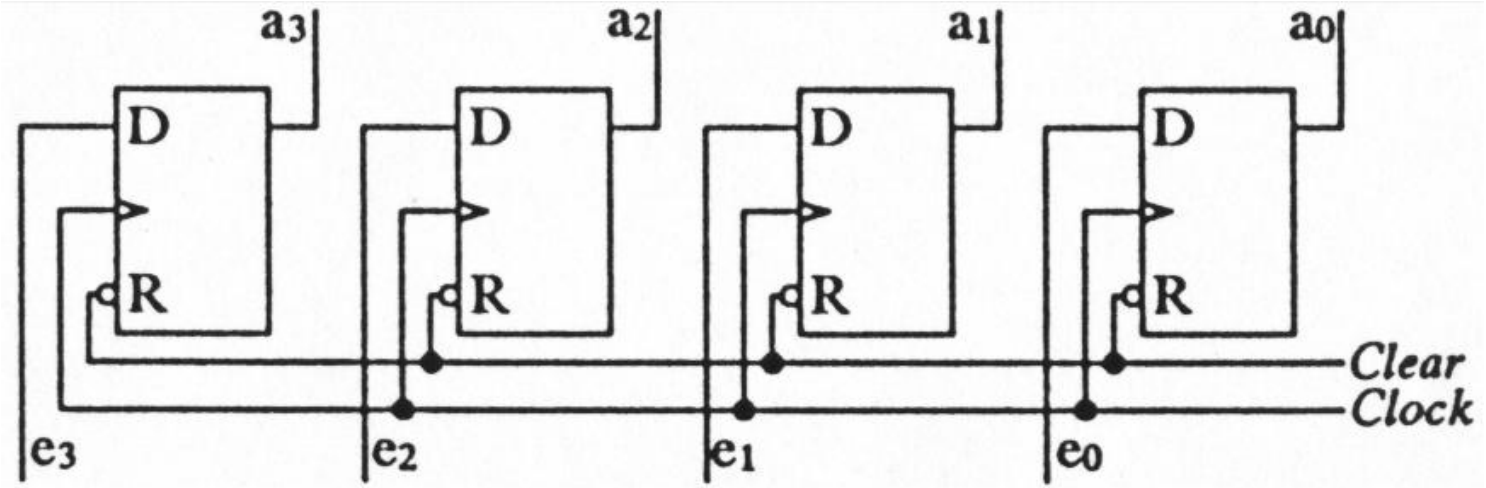
obvod prepínaný nábežnou hranou



obvod prepínaný zostupnou hranou

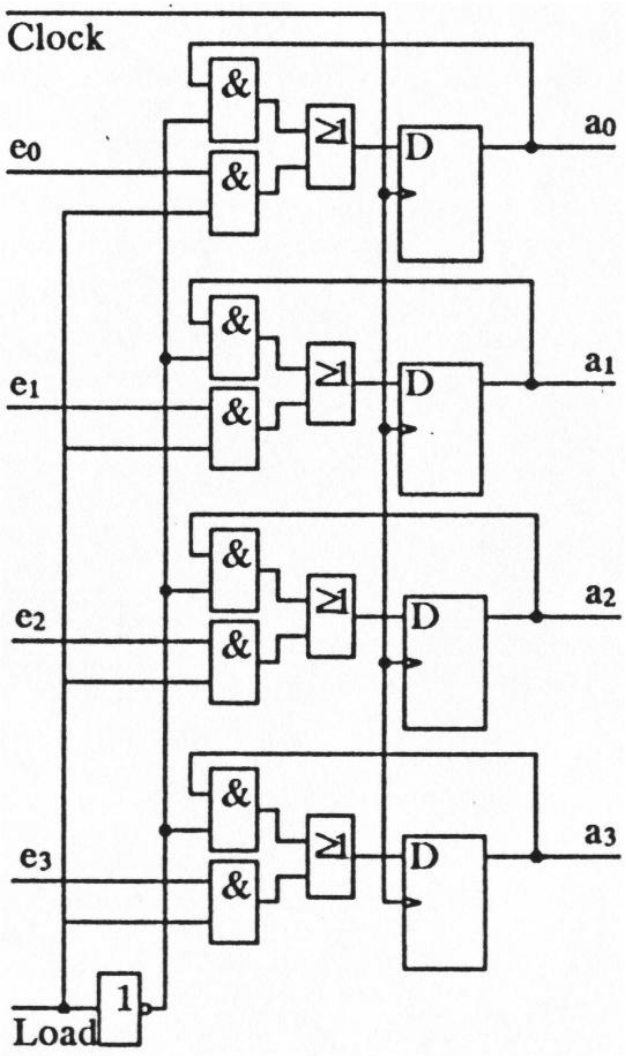
Register

- Jednoduchý s neštandartným D obvodom, s asynchrónnym R vstupom

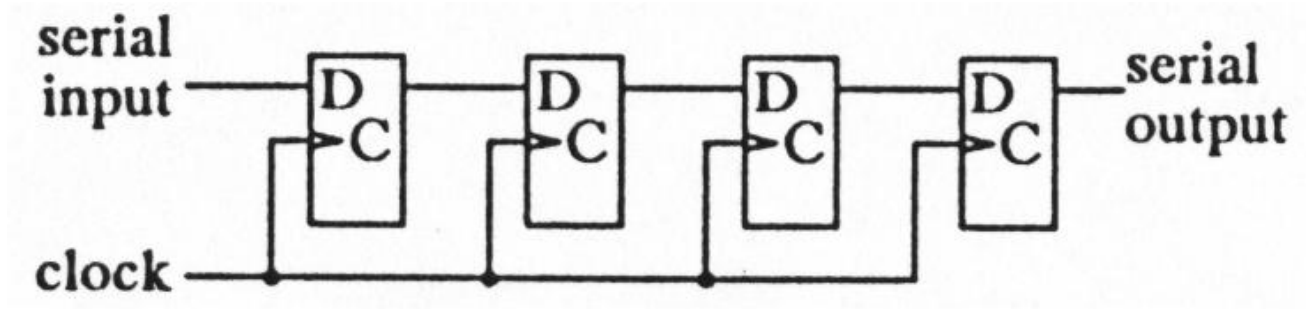


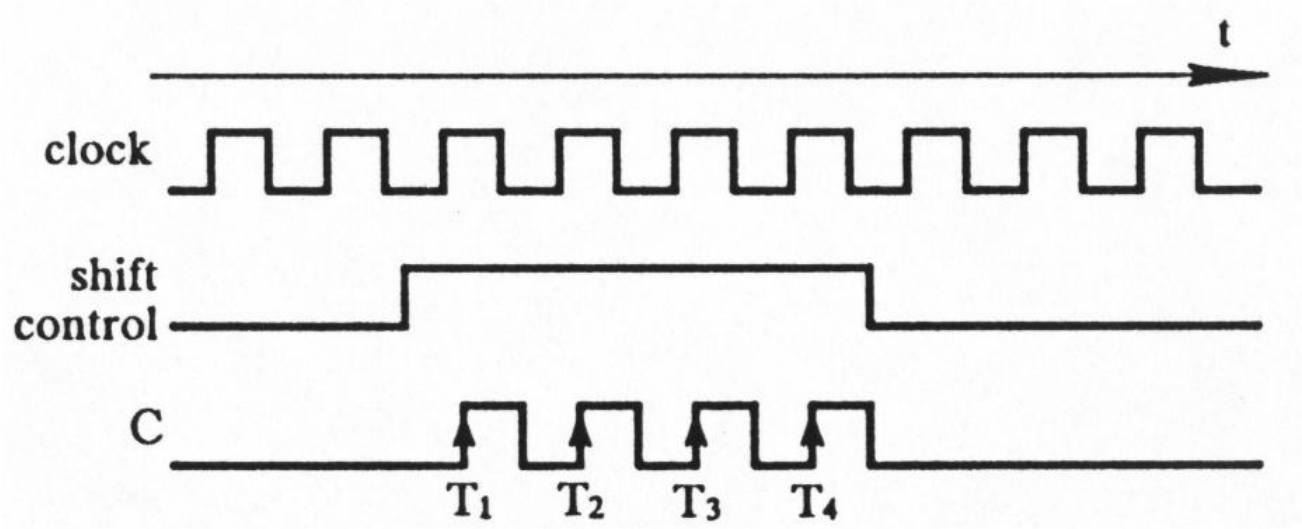
Register s Load

Ak Load=0, na výstupe je hodnota registrov
Ak Load=1, uloží sa hodnota zo vstupu



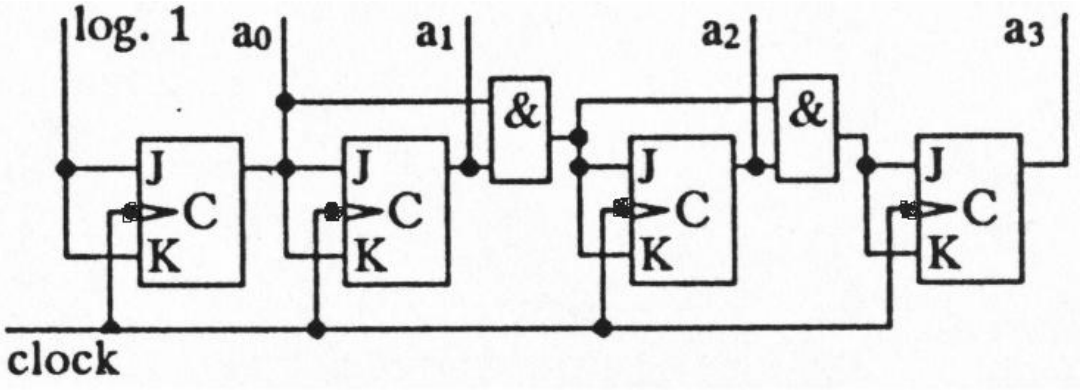
Sériový prenos, posuvný register

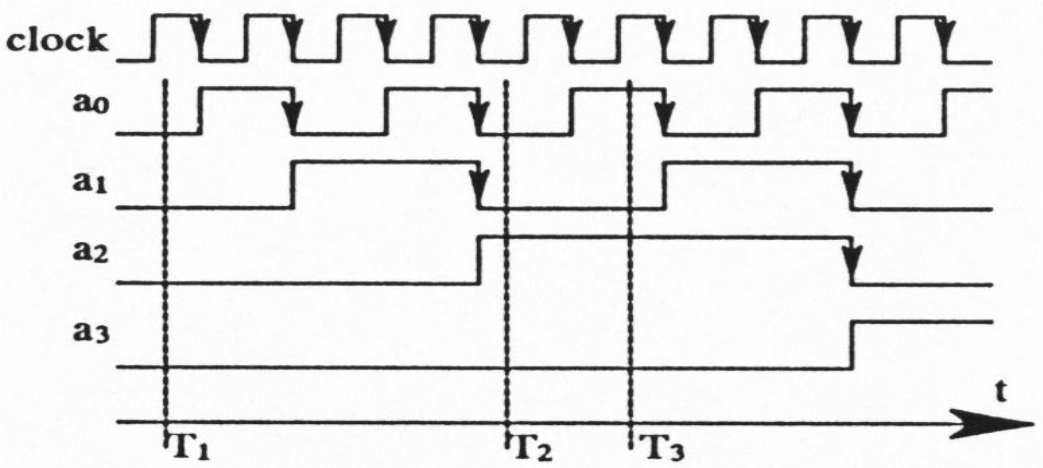




Čítač(Counter)

- Úlohou je počítať impulzy časovača

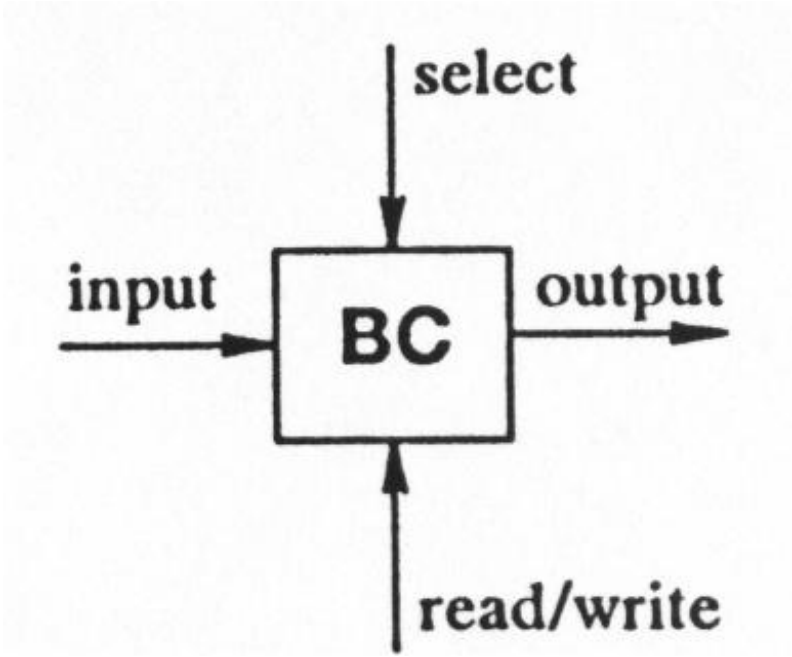


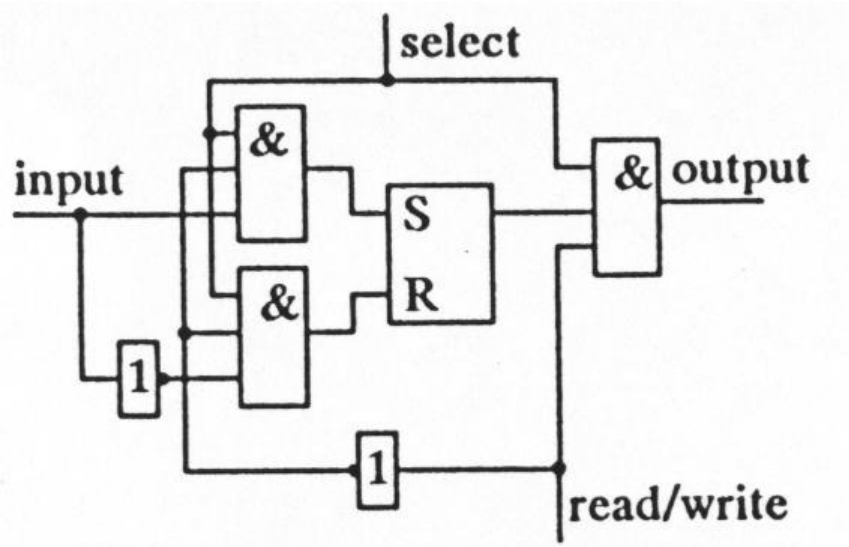


Pamäť

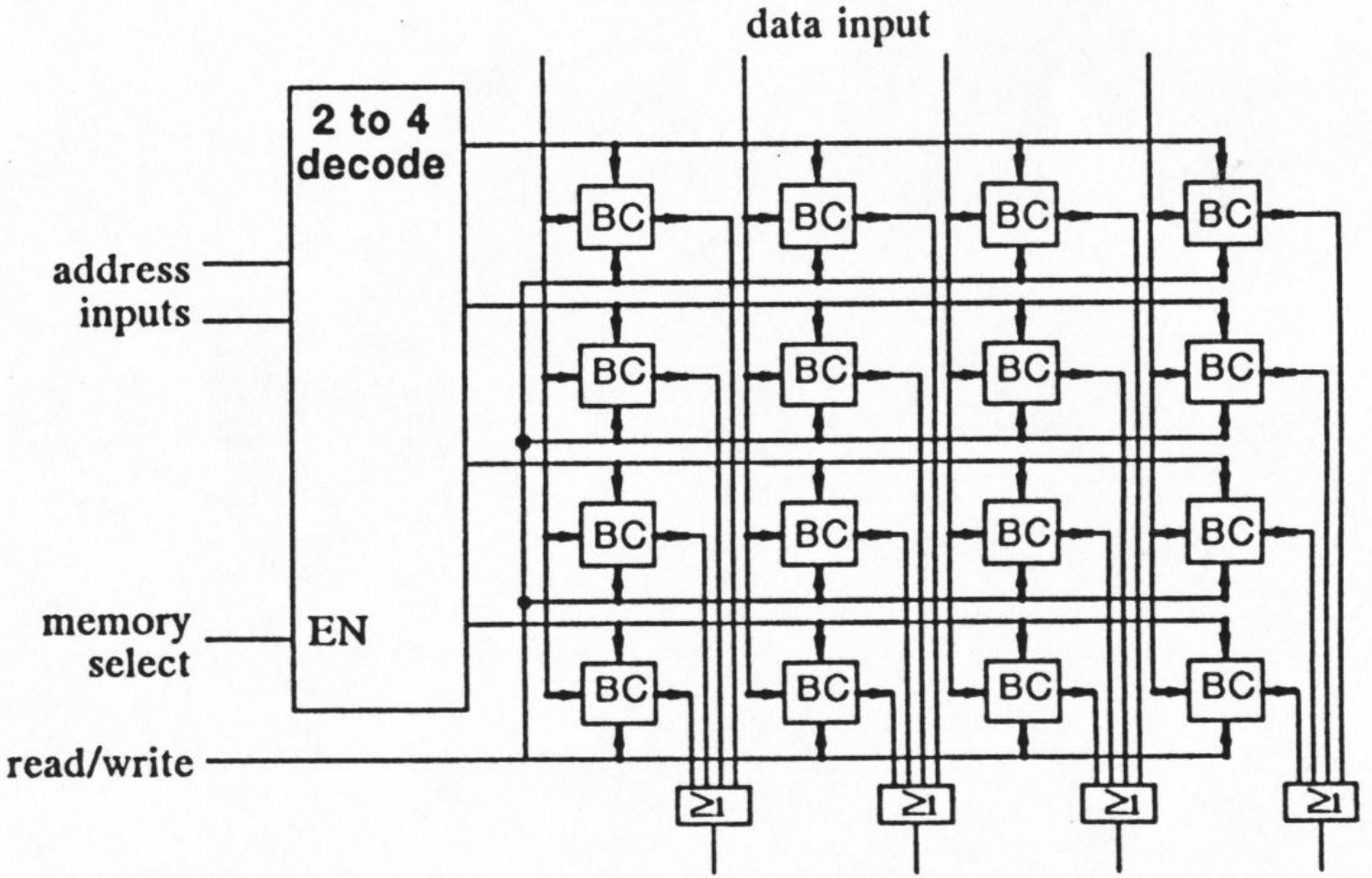
- RAM(random access memory)-môžeme pristupovať ku hociktorej položke
- RWM(read write memory)-môžeme čítať aj zapisovať
- Operačná pamäť je RAM, RWM
  - adresuje po riadkoch, v riadku je uložený určitý počet bitov

Pamäťová bunka





Pamäť 4x4



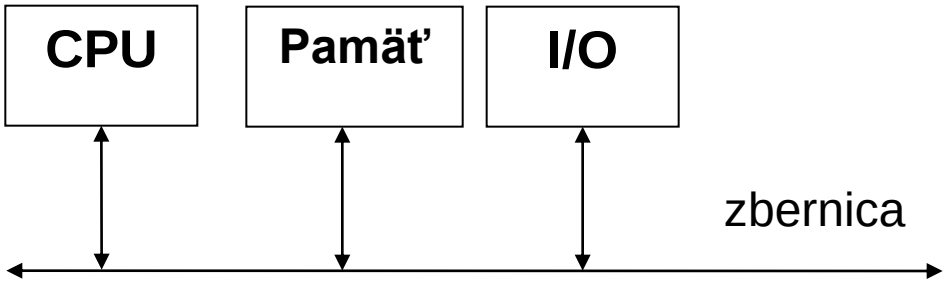
4x4 memory

Obsah

- Štruktúra počítača
- Funkcia a klasifikácia procesorov
- Schéma procesora
- Inštrukcie – úvod
- Registre
- Metódy adresácie argumentov

Štruktúra počítača von Neumannovského typu - schéma

- Počítač vykonáva činnosť podľa programu, ktorý je uložený v pamäti
- CPU – Central Processing Unit - riadi činnosť počítača a vykonáva inštrukcie
  - ALU – Arithmetical Logic Unit
  - CLU - Control Logic Unit
- Pomocou pamäte ukladá a uchováva informácie – programu aj údajov
- Pomocou vstupno/výstupných zariadení načítava vstup a zobrazí požadovaný výstup

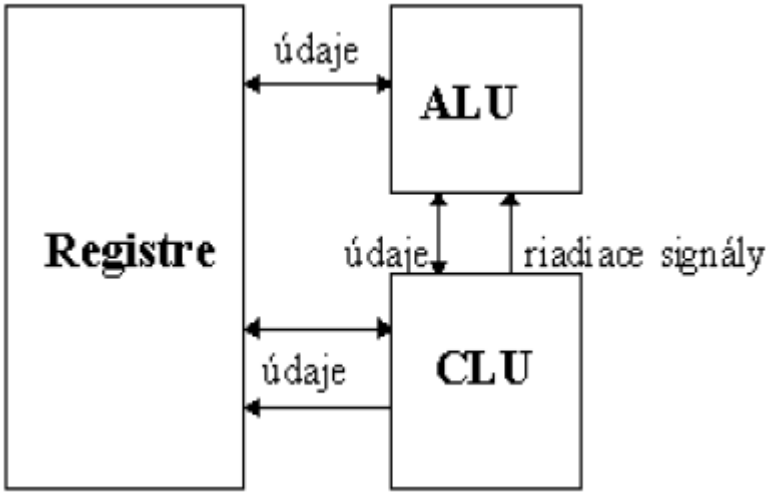


Funkcia a klasifikácia procesorov

- jadro číslicového počítača, realizuje hlavné operácie pri riadení a vykonávaní výpočtu, je definovaný programom
  - Univerzálne procesory
  - Problémovo-orientované
    - Aritmetický procesor – napr. FPP
    - Kanálový procesor – riadi I/O operácie
    - Videografický procesor

Schéma procesora

- Sada registrov
- Aritmeticko-logická jednotka – **ALU**
- Riadiaca jednotka – **CLU**
- Synchronizovaný pomocou časovača



Inštrukcia

- Organizácia počítača je určená množinou inštrukcií, ktoré počítač vykonáva
- Inštrukcia - binárny vektor, ktorý slúži na označenie operácie a jej operandov
- Množina inštrukcií, ktoré je schopný počítač vykonať - **inštrukčná sada** počítača
- **Formát inštrukcie**
  - Operačný kód určuje operáciu
  - Pole operandov
    - Konštanta, register, pamäťové miesto – určené adresou

Typy inštrukcií

- Presun dát medzi registrami, procesorom, pamäťou
- Aritmetické a logické operácie
- Presun dát z I/O zariadení
- Vetvenie programu
- Riadiacie inštrukcie

Inštrukcie presunu dát

- Priradenie hodnoty jedného operanda druhému
- Výmena hodnôt operandov
- Práca s blokmi údajov v pamäti – kopírovanie, presun, priradenie konštantnej hodnoty
- Operácie so zásobníkom – vkladanie, výber údajov do(zo) zásobníka

Aritmetické a logické operácie

- +,-,*,/
- Inc, dec
- Porovnávanie
- Zmena dĺžky
- BCD aritmetika
- Operácie na reálnych číslach
- Logické funkcie
- Posuvy, rotovania

Riadenie programu

- **Vetvenia, skoky, cykly**
  - skoky, absolútne i relatívne zadanie adresy skoku, podmienené skoky
  - cykly
  - podprogramy, príkaz pre skok do podprogramu, príkaz pre návrat z podprogramu
- **Prerušenia**
  - vyvolanie preršuenia
  - maskovanie prerušenia
  - vytvorenie podprogramu na obsluhu prerušenia a návrat z neho

Registre

- Množina registrov závisí od konkrétneho počítača
- Registre môžu mať rozličnú veľkosť
- Niektoré registre sa štandardne používajú v skoro všetkých univerálnych počítačoch
- **Program counter register** (PC)
  - Slúži na ukladanie adresy nasledujúcej inštrukcie, ktorá sa má vykonávať
  - Musí sa dať inkrementovať o +1
  - Musí sa doň dať uložiť adresa (číslo) pri operácii skoku

Registre(2)

- **Instruction register** (IR)
  - Slúži na uchovávanie inštrukcie, ktorá sa práve vykonáva
  - Dôvody:
    - na vykonanie inštrukcie bude možno potrebné čítať niekoľkokrát z pamäte; ak by vykonávaná inštrukcia bola uložená v MBR, pri ďalšom čítaní z pamäte by sa stratila
    - Spracovanie inštrukcie si vyžaduje špeciálny hardvér (napr. na dekódovanie operačného kódu)
- **Memory address register** (MAR, register adries pamäte)
  - Slúži na ukladanie adresy pamäťového miesta, s ktorým sa má pracovať
  - Jeho veľkosť je určená veľkosťou pamäte, ktorú má adresovať
  - Register s paralelným zápisom a čítaním
- **Memory buffer register** (MBR, vyrovnávací register pamäte)
  - Prostredníctvom neho sa čítajú údaje z pamäte a zapisujú sa do pamäte
  - Jeho veľkosť je rovnaká ako veľkosť pamäťového miesta
  - Register s paralelným zápisom a čítaním

Registre(3)

- **Accumulator** (ACC, akumulátor)
  - Jeden z operandov aritmetickej a logickej operácie je (skoro vždy) obsah akumulátora
  - Výsledok operácie sa ukladá do akumulátora
  - Takéto riešenie zjednodušuje výkonné aj riadiace obvody
  - Niektoré počítače používajú ako akumulátor jeden z univerzálnych registrov (pseudo-ACC)

Príznakové registre(Flag registers)

- pomocou nich sa kódujú rôzne udalosti, ktoré vznikli počas behu programu
- na ich základe dochádza k vetveniu programu
- Aritmetické príznaky informujú o výsledku naposledy vykonanaj aritmetickej operácie

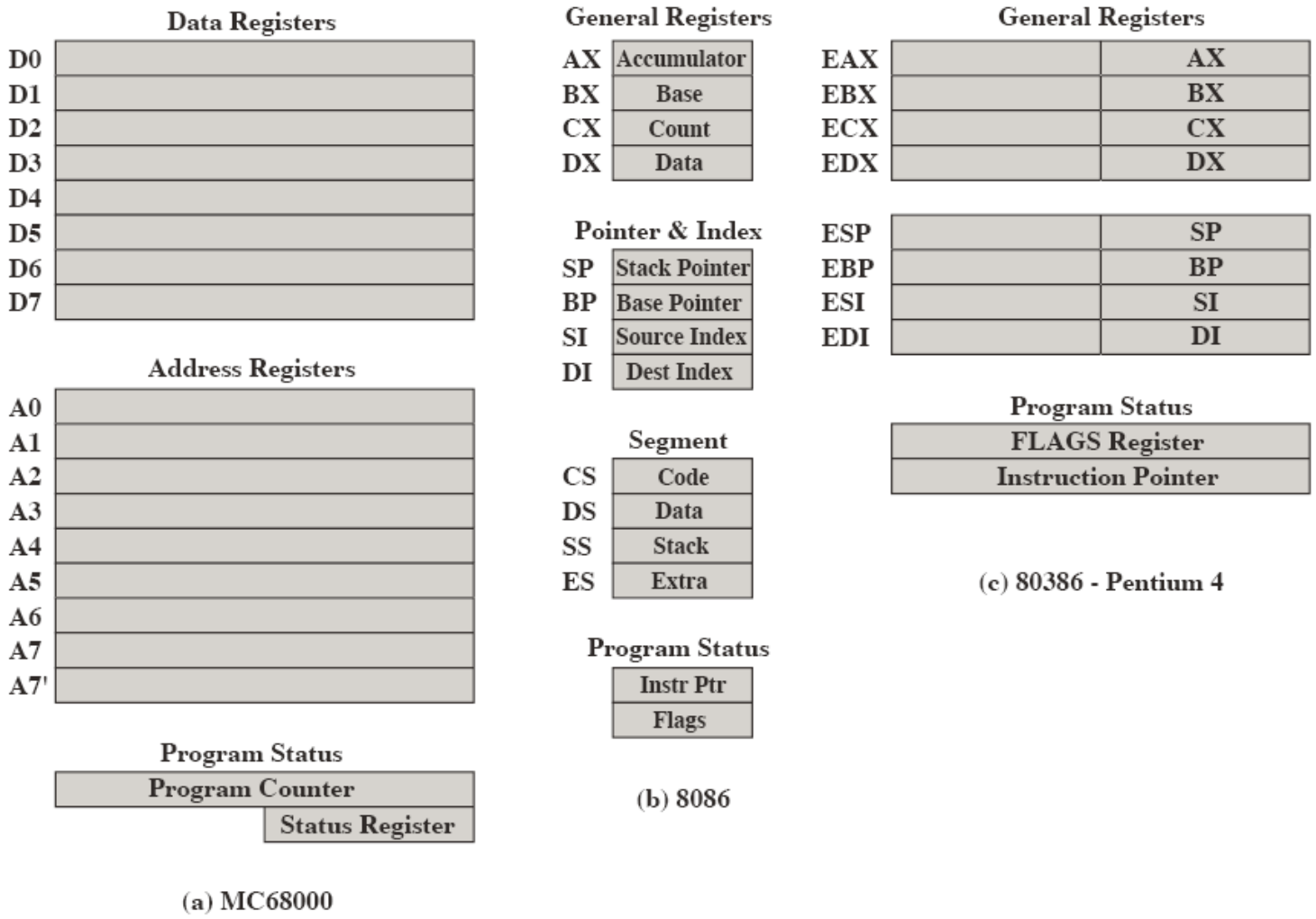
Príznakové registre (2)

- **CF - Carry Flag**
  - príznak pretečenia neznamienkových čísel
- **OF - Overlow Flag**
  - pretečenie znamienkových čísel
- **SF - Sign Flag**
  - znamienko čísla kópia najvyššieho bitu čísla
- **ZF - Zero Flag**
  - príznak nuly - výsledok je nulový
- **AC - Auxiliary Carry Flag**
  - pretečenie pri BCD číslach
- **PF - Parity Flag**
  - parita výsledku

Ďalšie registre

- **Program Status Word (PSW)**
  - Obsahuje informáciu o stave CPU
    - príznaky, detakcia prerušení, vnútorné stavy
- **Stack Pointer SP**
  - Vrchol zásobníka
  - Zásobník je pamäťová dátová štruktúra...

Príklad organizácie registrov



Metódy adresácie argumentov

- Určujú, ako treba interpretovať adresové pole inštrukcie; odkiaľ brať operandy
- operand:
  - Konštanta
  - Obsah registra
  - Obsah pamäťového miesta

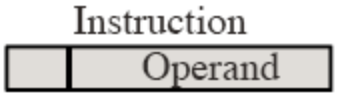
Metódy adresácie argumentov(2)

Mód	Hodnota operandu	Označenie
Implicitný	Žiadna	
Bezprostredný	konštanta	OPR:=číslo
Priamy	Pamäť na adrese	OPR:=M[ADR]
Nepriamy	Pamäť na adrese adresy	OPR:=M[M[ADR]]
Register	Obsah registra	OPR:=(R1)
Register (nepriamy)	Pamäť na adrese z registra	OPR:=M[(R1)]
Autoinkrement	Register, inkrement reg.	OPR:=(R1); ++R1
Relatívny	Pamäť na zloženej adrese	OPR:=M[(PC)+ADR]
Indexový	Pamäť na zloženej adrese	OPR:=M[(IX)+ADR]
Bázový	Pamäť na zloženej adrese	OPR:=[(BS)+(R1)]

Metódy adresácie(2)

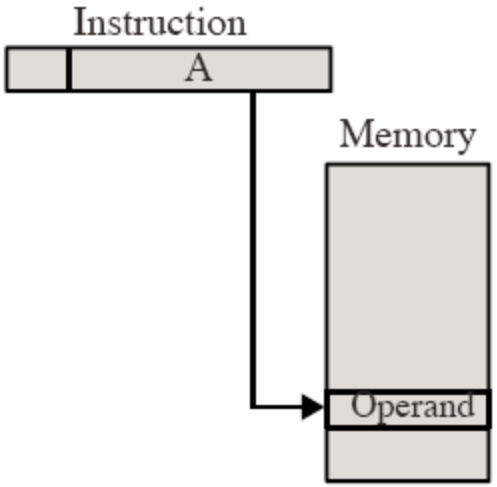
- **Implicitný mód (implied mode)**
  - Operandy sú špecifikované priamo v operačnom kóde
  - implicitným operandom aritmetickej operácie je obsah akumulátora
  - skok na pevne pevne danú adresu so špeciálnym významom
  - implicitnú adresáciu využíva zásobník
- **Bezprostredný spôsob adresovania (immediate addressing mode)**

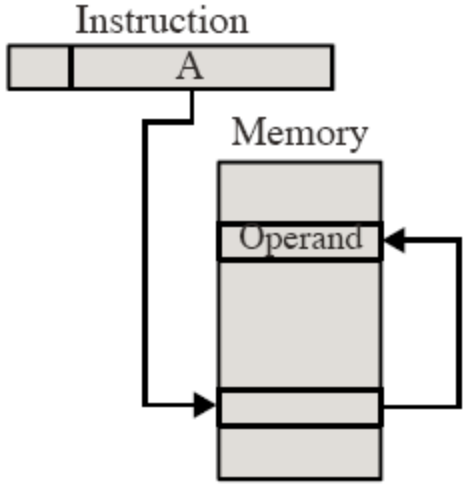
V adresovom poli sú uložené konštanty (tento spôsob adresovania využíva napríklad operácia SHIFT)



Metódy adresácie(3)

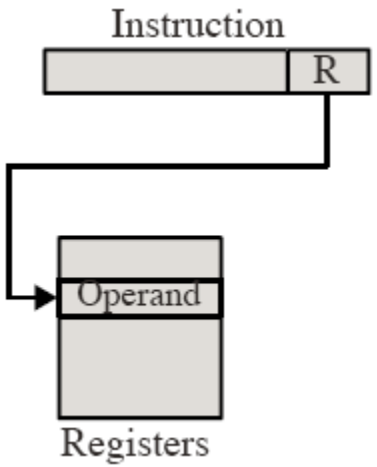
- Priame a nepriame adresovanie (direct and indirect addressing modes)
- Jeden bit slúži na odlíšenie, či ide o priamu, alebo nepriamu adresáciu

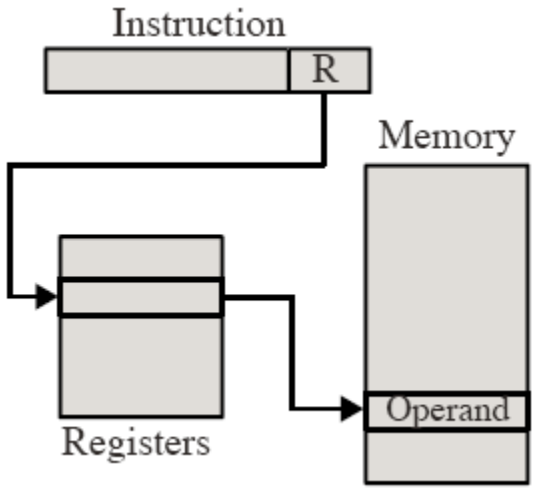




Metódy adresácie(4)

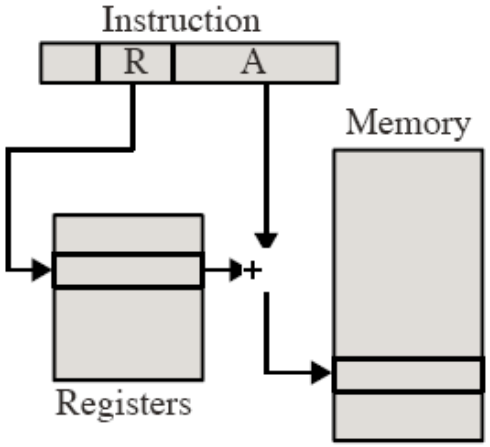
- Register priamy a nepriamy





Zložitejšie metódy adresovania

- Relatívny
  - OPR:=M[(PC)+ADR]
  - obvykle k adrese danej inštrukcie sa pipočíta adresa ADR
- Indexový
  - OPR:=M[(IX)+ADR]
  - jeden register sa využíva ako indexový
  - adresa=obsah adresového poľa + (IX)
- Bázový
  - OPR:=[(BS)+(R1)]
  - adresa operandu sa počíta ako súčet obsahu
    - bázového registra BS (základná adresa)
    - obsahu adresového poľa (R1)
- Bázový+indexový



Zložitejšie metódy adresovania(2)

- Zreťazenie
  - OPR:=(R1)(R2)
- Blokové adresovanie
  - využíva adresu na určenie pozície prvého slova v bloku údajov (pásky, disky)
  - blok má pevnú, alebo premenlivú dĺžku
    - Pri blokoch premenlivej dĺžky potrebujeme určiť koniec bloku:
      - Adresa začiatku a konca bloku
      - Adresa začiatku a údaj o dĺžke bloku
      - EOB znak detekujúci koniec

Obsah

- Princíp vykonávania inštrukcií
- Inštrukčný cyklus
- RTL
- Fetch
- Indirect
- Execute
- Prerušenie
- Mikroinštrukcie

Princíp vykonávania inštrukcií

- Inštrukcia je uložená v pamäti počítača
- Vykonanie operácie, ktorú inštrukcia určuje, sa nedá uskutočniť v jednom takte
  - Napr. sa musí preniesť z pamäte do registra
- Počítač vykonáva inštrukcie pomocou postupnosti elementárnych operácií, ktoré sa nazývajú mikrooperácie
- Mikrooperácie sú určené mikroinštrukciami
- Mikrooperácie sa dajú vykonať v jednom takte
- Postupnosti mikroinštrukcií sa nazývajú mikroprogramami (alebo CPU cyklami)

Inštrukčný cyklus

- Pri spracovaní inštrukcie sa uplatňujú nasledujúce CPU cykly:
  - **Fetch**
    - Načítanie inštrukcie z pamäte
    - Načítanie operandov v prípade nepriamej adresácie(**indirect**)
      - Address (dekódovanie adresy operandu)
      - Translation
  - **Execute** - vykonanie inštrukcie, zápis výsledkov
  - **Interrupt** - ošetrenie prerušenia, nastane iba ak systém zaznamenal požiadavku o prerušenie
- Mikrooperácie spôsobujú prenos údajov medzi registrami, preto ich výhodne možno popisovať pomocou RTL (register transfer language), ktorý sme používali pri návrhu digitálnych systémov

Register Transfer Language (1)

- Zápis do registra

  B := (A)

- Môžu sa prenášať aj časti registrov (polia)

  PC := IR[AD]

- Ak časť registra nemá meno, tak

  R1[0..3] := (X)

- Registrom sa dajú priraďovať konštanty

  L := 5

Register Transfer Language (2)

- Aritmetické operácie

  A3 := (A1) + (A2); súčet
  A := (A) + 1; inkrementovanie obsahu
  A := (A) - 1; dekrementovanie
  A := (~A); logický doplnok
  A := (~A) + 1; binárny doplnok
  A := (A) +(~B) + 1; odčítanie A-B
- Logické operácie

  C := (A) AND (B); logický súčin
  C := (A) OR (B); logický súčet
- Operácie posunu

  A := SL(A); posun doľava
  A := SR(A); posun doprava
  A := LCIR(A); cyklický posun doľava
  A := RCIR(A); cyklický posun doprava

Register Transfer Language (3)

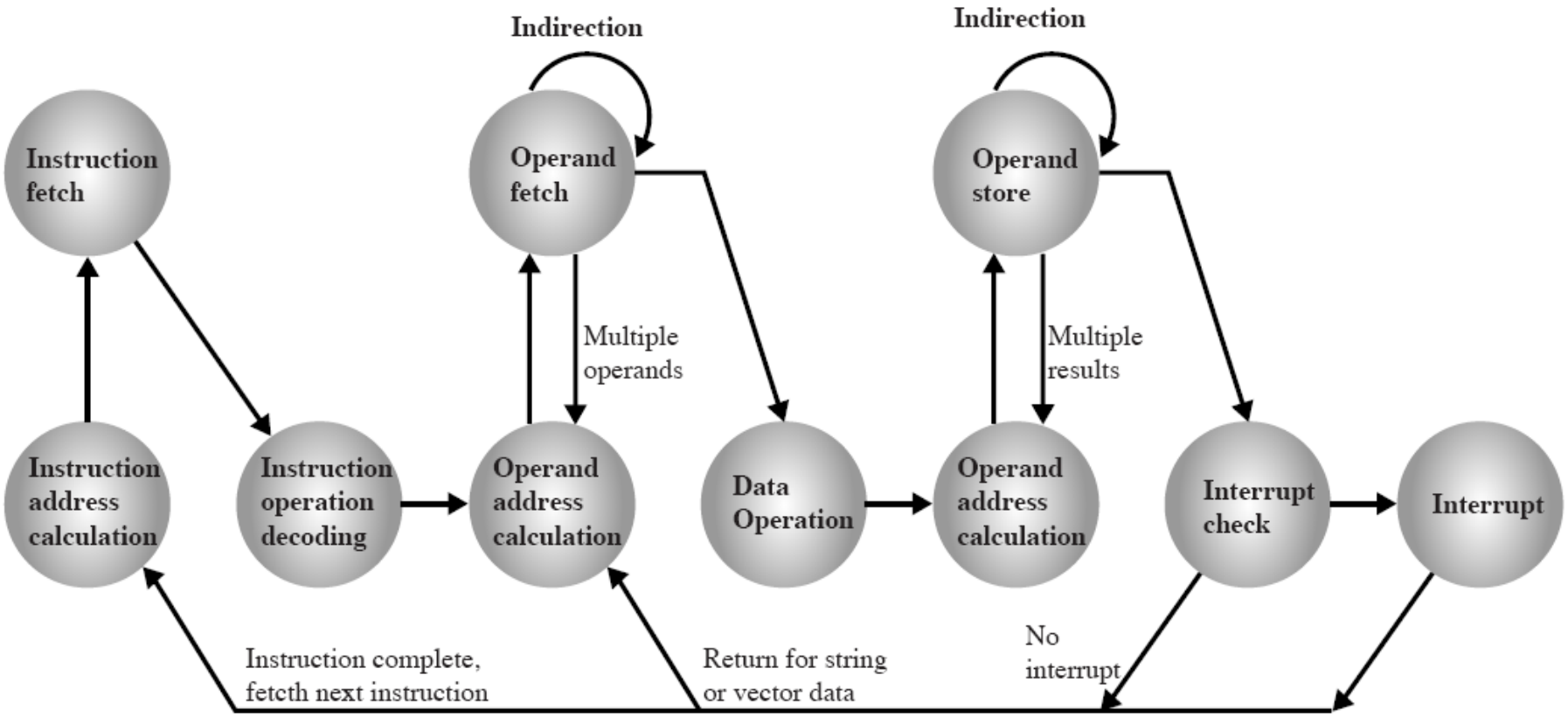
- Operácie presunu informácie medzi registrami a pamäťou
- Čítanie z pamäte

  B := (M[(A)]); obsah pamäťového miesta, ktorého adresa je v registri A sa zapíše do registra B
- Zápis do pamäte

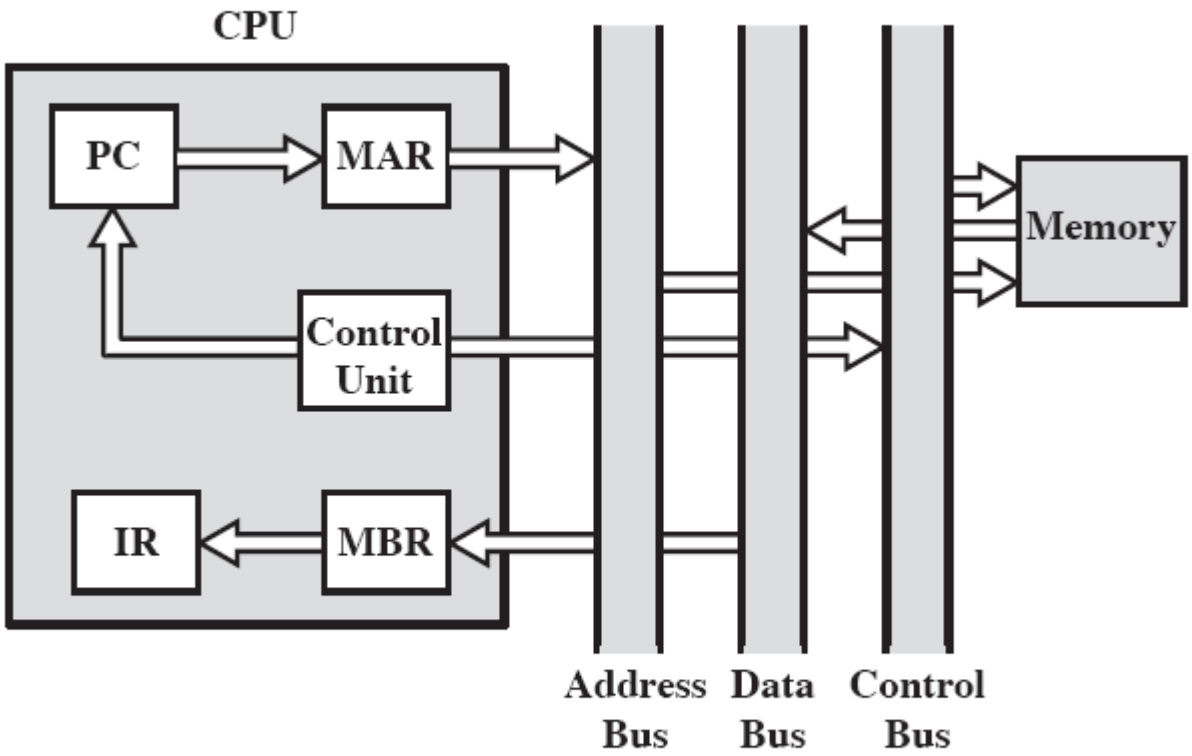
  M[(A)] := (B); do pamäťového miesta, ktorého adresa je v registri A sa zapíše obsah registra B
- Vykonávanie operácií sa niekedy viaže na splnenie nejakých podmienok:
  - Logické podmienky: **IF ... THEN ...**
  - Riadiace podmienky = logické funkcie definované na Booleovských premenných, ktoré riadia vstupy do registrov

    $t_0(c_1+c_2)$: X := (~A) + 1

Inštrukčný cyklus(2)

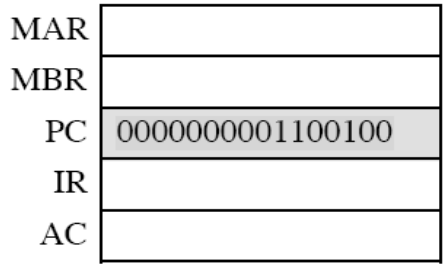


Tok dát pri fetch

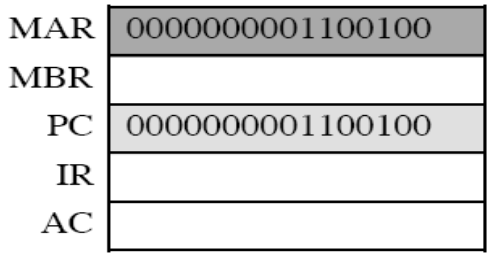


MBR = Memory buffer register
MAR = Memory address register
IR = Instruction register
PC = Program counter

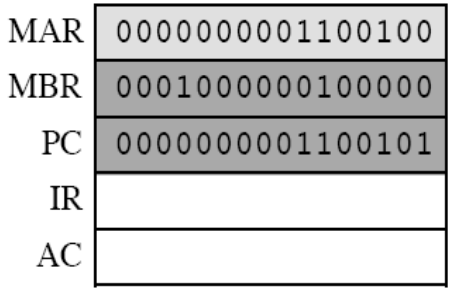
Príklad fetch cyklu



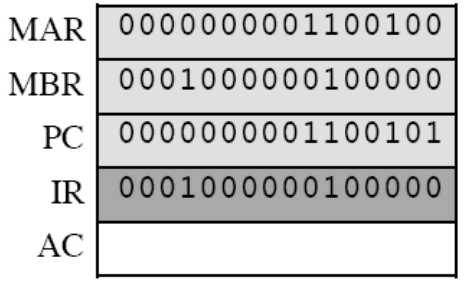
- t_1: MAR:=(PC)



- t_2: MBR:=(M[(MAR)])
  PC:=(PC)+1



- t_3: IR:=(MBR)



Indirect

- Predpokladajme, že inštrukcia má jeden operand s nepriamou adresáciou
- t_1: MAR:=(IR[ad]); ad je úsek inštrukcie, kde sa nachádza adresa operandu
- t_2: MBR:=(M[(MAR)])
- t_3: IR[ad]:=(MBR)
- Teraz IR neobsahuje nepriame adresy

Execute(1)

- ADD R1, X – pripočíta ku registru R1, čo je uložené na adrese X
- t_1: MAR:=(IR[ad]); ad je úsek inštrukcie, kde sa nachádza adresa operandu t.j X
- t_2: MBR:=(M[(MAR)])
- t_3: R1:=(R1)+(MBR)

Execute(2)

- ISZ X – increment and skip if zero
  - Obsah, čo je uložený na adrese X je inkrementovaný o 1, ak výsledok je 0, nasledujúca inštrukcia je vynechaná
- t_1: MAR:=(IR[ad])
- t_2: MBR:=(M[(MAR)])
- t_3: MBR:=(MBR)+1
- t_4: M[MAR]:=(MBR)

  IF (MBR)=0 THEN PC:=(PC)+1

Execute(3)

- BSA X – branch and save
  - Na adrese X sa uloží adresa, ktorá nasleduje po tejto inštrukcii a program pokračuje na adrese X+1, uložená adresa poslúži pre návrat
- t_1: MAR:=(IR[ad])

  MBR:=(PC)
- t_2: PC:=(IR[ad])

  M[MAR]:=(MBR)
- t_3: PC:=(PC)+1

Prerušenie

- Procesor umožňuje prerušiť prebiehajúcu činnosť, začať vykonávať inú a potom sa vrátiť ku pôvodnej
- Prerušenie nastáva, ak chce zariadenie prinútiť procesor, aby sa s ním zapodieval
- Zariadenie vyšle žiadosť, po prijatí si procesor zapamätá svoj vnútorný stav a začne vykonávať obslužný program

Prerušenie(2)

- Procesor jednoznačne identifikuje žiadateľa (napr. číslo 0..255),
- Má tabuľku obslužných programov resp. adries
- Nemusí ísť len o “vonkajšie” volanie, ale aj niektoré vnútorné udalosti napr. delenie nulou
- Zariadenie má svoju prioritu
- Rozlišujeme maskovateľné a nemaskovateľné prerušenia. Maskovateľné sa dajú zakázať, nemaskovateľné nie.

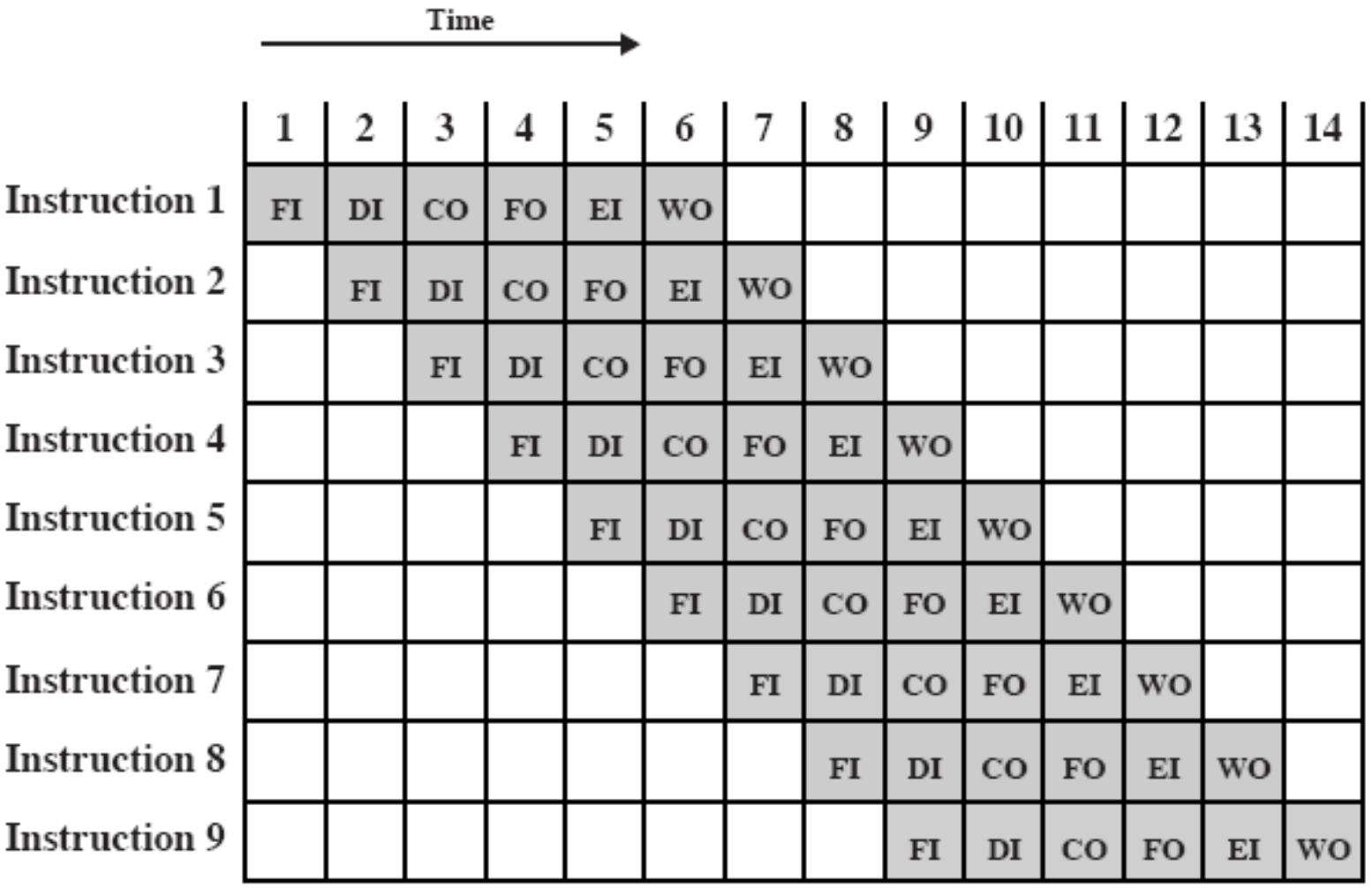
Prerušenie(3)

- Uchová aktuálny stav regsitrov
- Z viac žiadostí vyberie s najvyššou prioritou
- Zistí adresu obslužného programu
- Odovzdá mu riadenie
- Po ukončení sa vráti ku pôvodnej činnosti

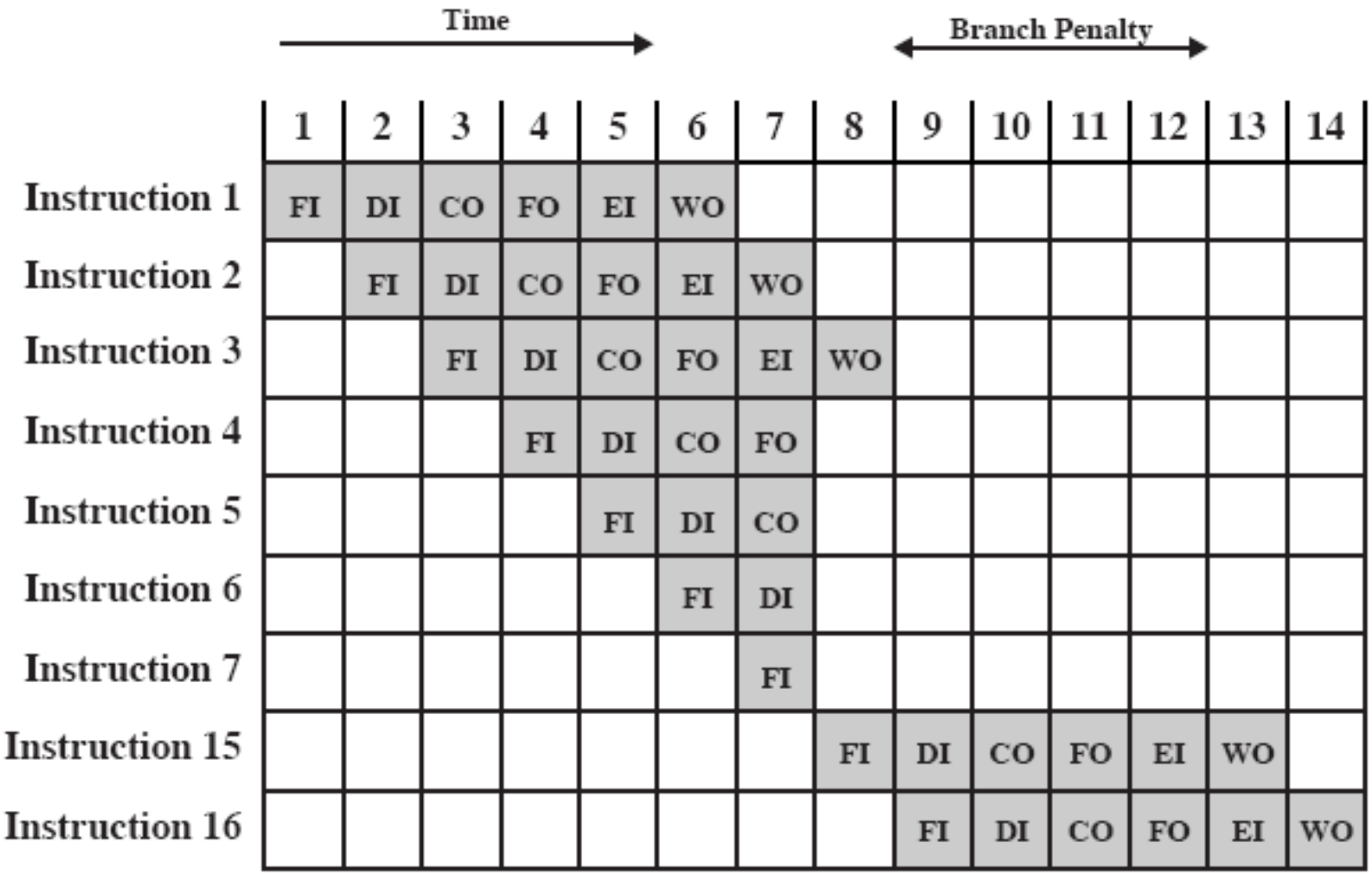
Pipelining inštrukcií- prúdové spracovanie

- Dekompozícia inštrukcie
  - Fetch instruction FI
  - Decode instruction DI
  - Calculate operands CO
  - Fetch operands FO
  - Execute instruction EI
  - Write operand WO
- Vykonávanie ďalších inštrukcií paralelne
- Problémy
  - EI môže trvať dlhšie
  - Vetvenie programu
  - Problémy s konfliktom pamäte

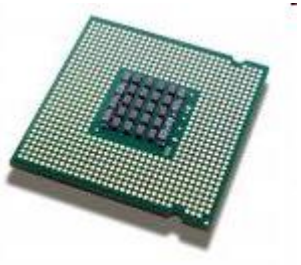
Pipelining – ideálny stav



Pipelining(3)



Obsah

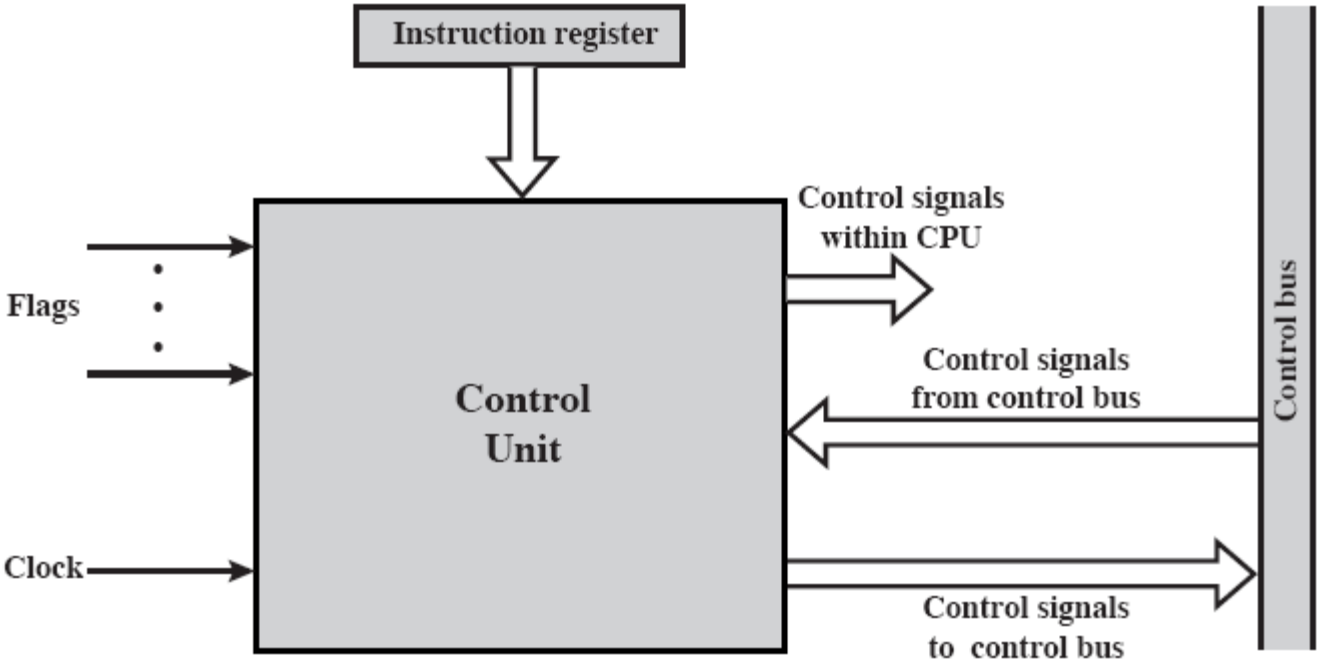


- CLU
- Inštrukčný cyklus
- Kontrolné signály
- Príklad procesora: Intel 8085
- Pevne zapojená(hardwired) implementácia
- Implementácia mikroprogramovou logikou
- Výhody resp. nevýhody
- Zvyšovanie výkonu procesora
- RISC vs. CISC

Základné funkcie CLU

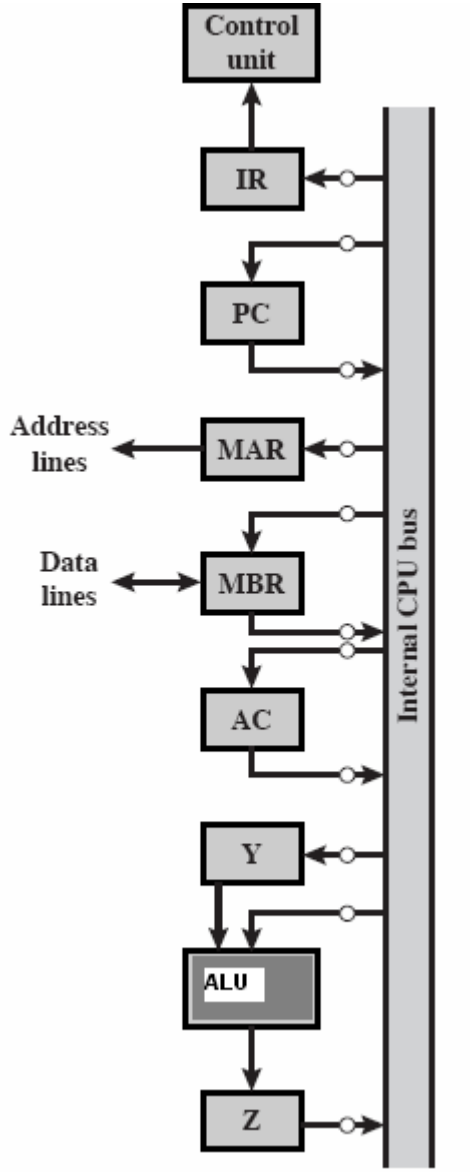
- Riadiaca jednotka, určená na generovanie vnútorných procesorových signálov a na vyhodnotenie stavových signálov o procesoch. Realizovaná
  - Napevno
  - Mikroprogramovateľnou logikou
- Synchronizačná jednotka je určená na časovanie jednotlivých činností procesora a systému. Základ je generátor hodinových impulzov.
- Inštrukčná jednotka je určená na výber inštrukcie, dekódovanie, prípravu a na vykonanie príslušnej operácie.
- Radič požiadaviek prerušenia

Model CLU


Instruction register
Control signals
within CPU
Flags
Control
Unit
Control signals
from control bus
Control bus
Clock
Control signals
to control bus

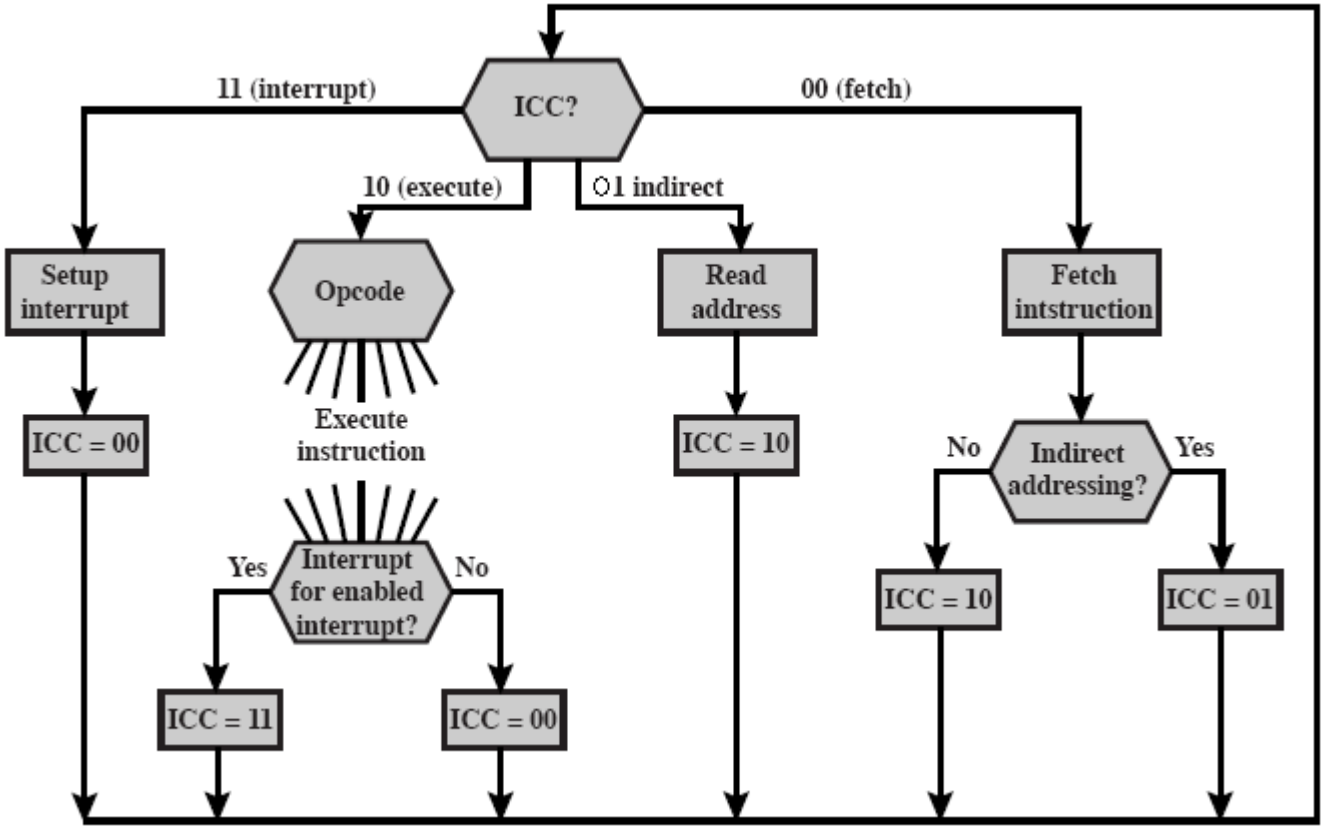
Model CPU

Y,Z registre sú pomocné napr.
pre vykonanie operácie s 2 operandmi

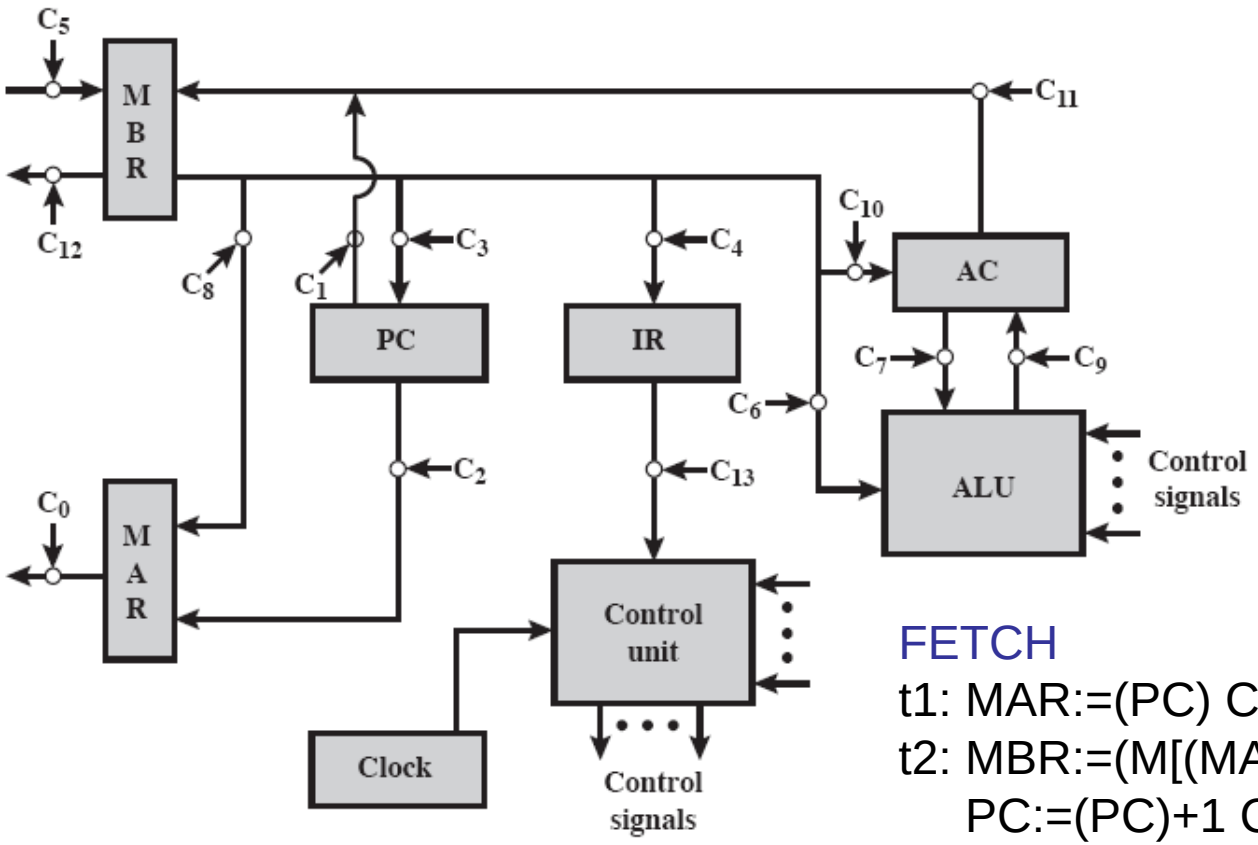


Inštrukčný cyklus

- ICC(instruction cycle code)
  - 00: Fetch
  - 01:Indirect
  - 10:Execute
  - 11:Interrupt



Riadiace signály



FETCH

t1: MAR:=(PC) C_2

t2: MBR:=(M[(MAR)]) C_5

PC:=(PC)+1 C_{PC+1}

t3: IR:=(MBR) C_4

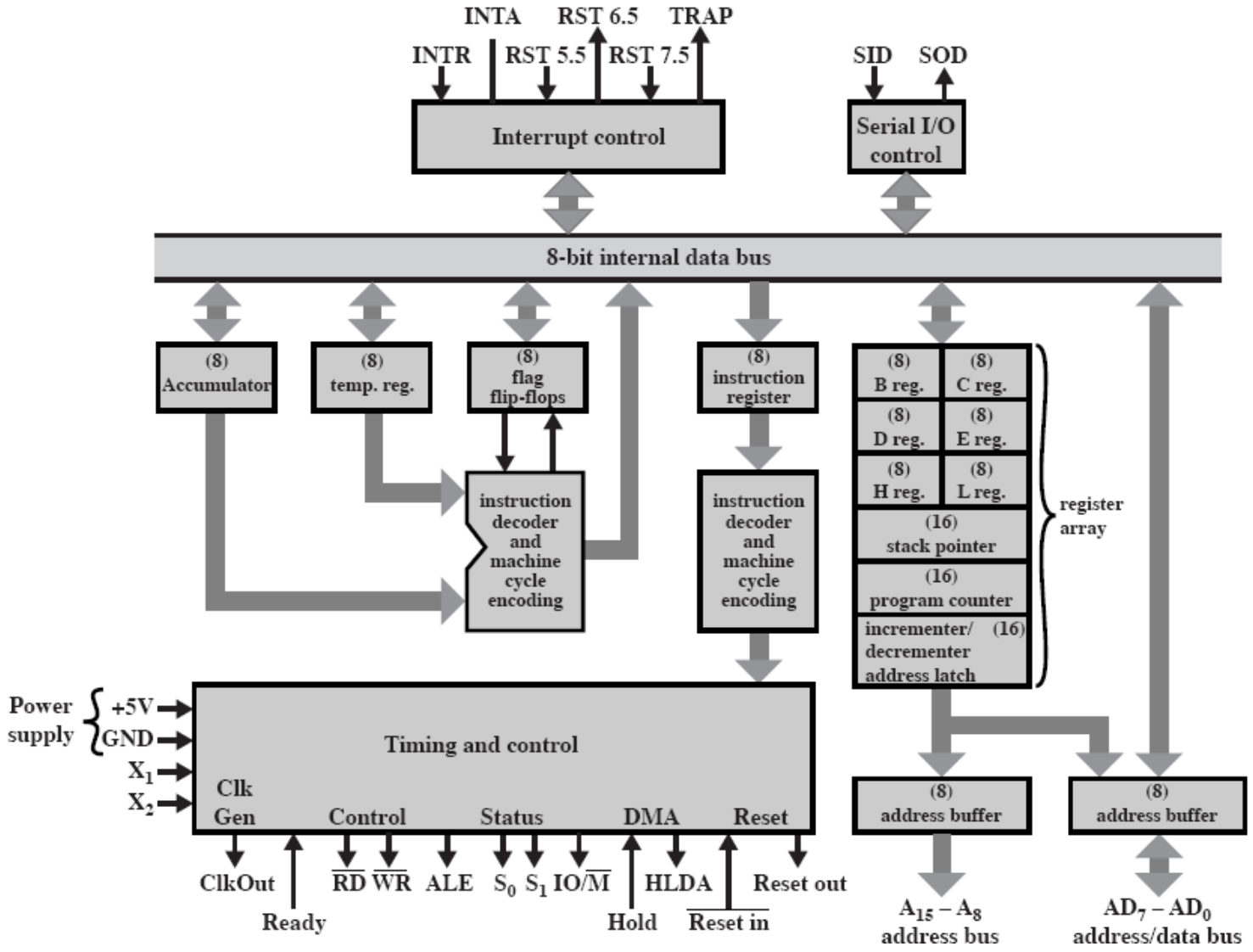
INDIRECT

t1: MAR:=(IR[ad]); C_8

t2: MBR:=(M[(MAR)]) C_5

t3: IR[ad]:=(MBR) C_4

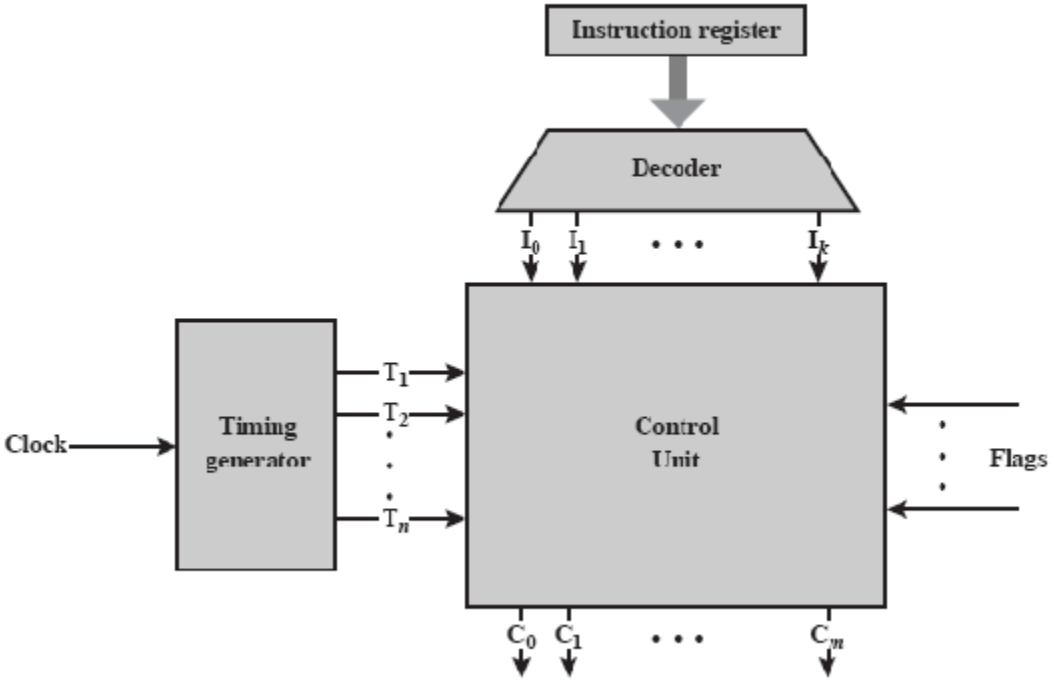
Príklad procesora: Intel 8085



Príklad procesora: Intel 8085(2)

- CLK - systémový čas
- ALE – vyskytuje sa na začiatku cyklu
- X1,X2 – signály z externého krištáľu pre vedenie interného generátora
- RD,RW – indikujú režim práce s pamäťou, I/0
- RESET IN – zapríčiňuje reset procesora
- RESET OUT – potvrdzuje, že CPU bolo resetnuté, signál sa používa pre reset systému

Pevne zapojená(hardwired) implementácia



PQ=00 Fetch
PQ=01 Indirect
PQ=10 Execute
PQ=11 Interrupt

$C_5 = P'.Q'.t_2 + P'.Q.t_2$

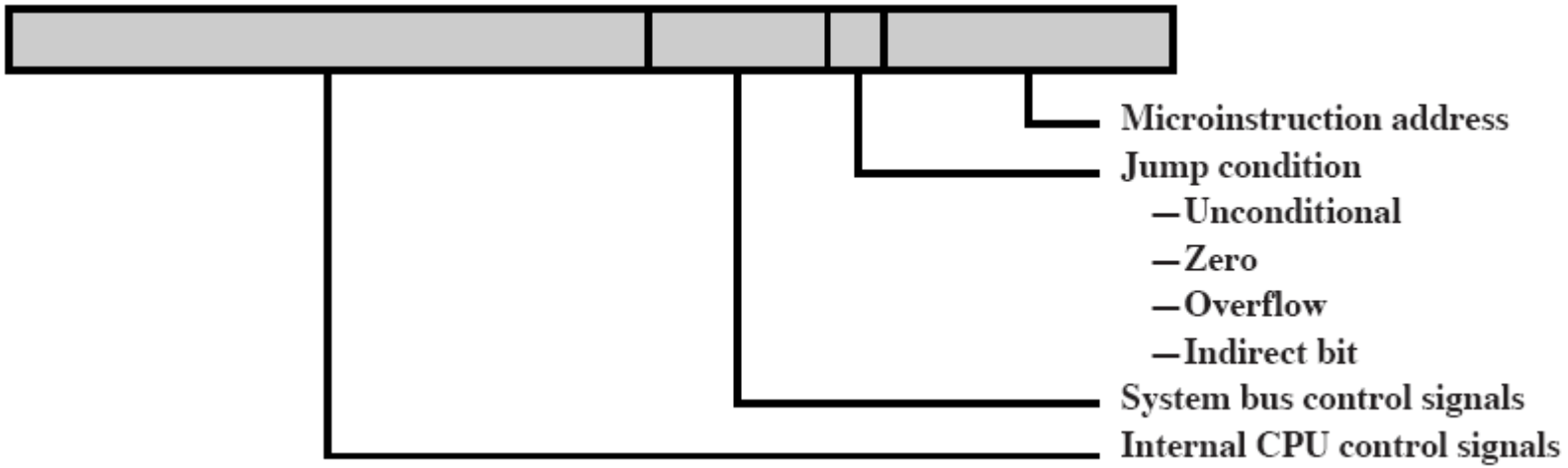
- Pre každý riadiaci signál je jeden obvod
- Nutné pre každú inštrukciu
- Rýchlejšie, avšak obmedzený, menší počet inštrukcií
- RISC architektúra

Implementácia mikroprogramovou logikou

- Riadenie vykonávania inštrukcie nie je riadené hardvérom, ale sa zapíše mikroprogramom (firmware)
  - Nemenné mikroprogramy
  - Možné čiastočné zmeny
  - Možno úplne naprogramovať užívateľom
- Typické pre CISC architektúru
- Veľký počet inštrukcií, avšak pomalšie ako pri hardwired realizácii

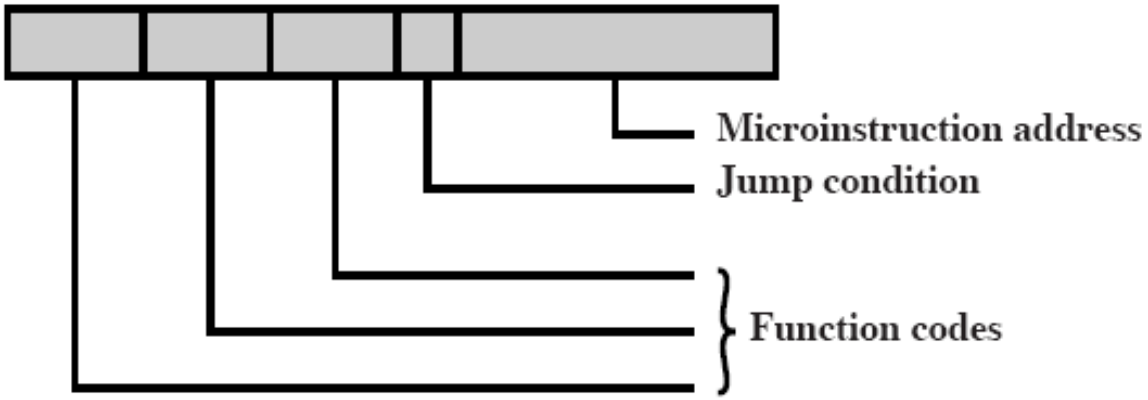
Horizontálny formát mikroinštrukcie

•Binárny vektor
•Obsahuje bity pre všetky riadiace signály t.j. ak je 1 tak vyšle signál, ak 0 tak nie
•Ak podmienku pri skoku vyhodnotí
    •false, tak pokračuje ďalšou mikroištrukciou v poradí
    •true, skočí na adresu uvedenú v mikroinštrukcii

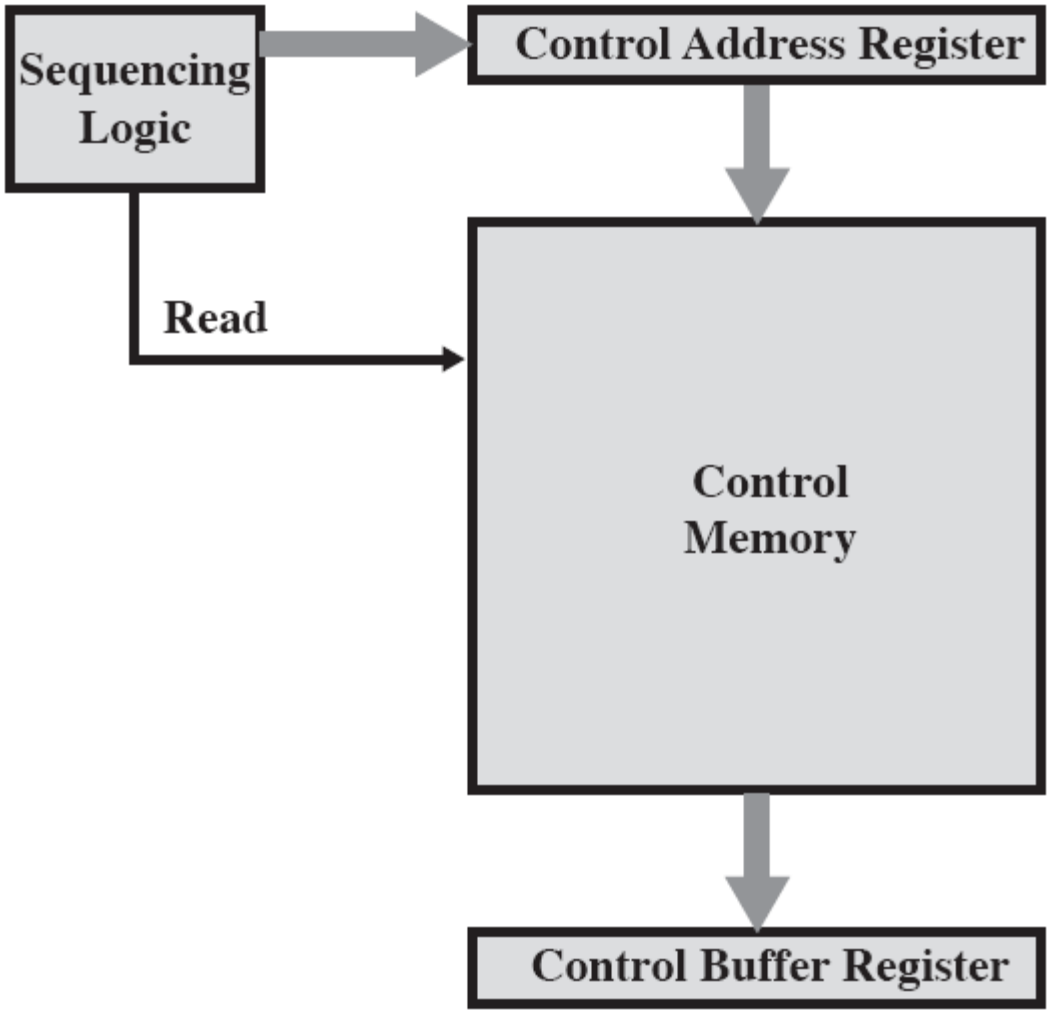


Vertikálny formát mikroinštrukcie

- Špecifikuje sa len jedna mikrooperácia
- Kratší zápis, dekódujú sa riadiace signály



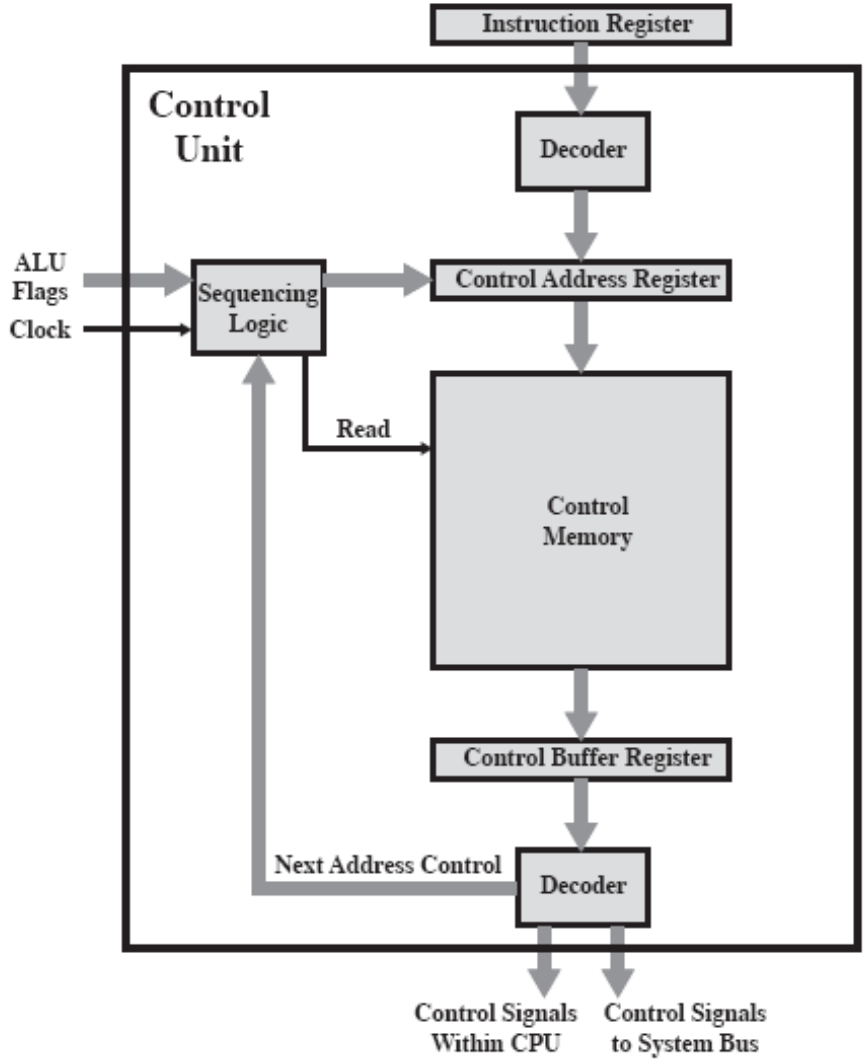
Realizácia CLU mikroprogramovateľnou jednotkou



Realizácia CLU mikropromovateľnou jednotkou(2)

- V Control Memory je uložená množina mikroinštrukcií
- Control Address Register obsahuje adresu ďalšej vykonávanej inštrukcie
- V control buffer register je uložená práve vykonávaná mikroinštrukcia
- Pri vykonávaní inštrukcie
  - CBR:=(CM[CAR]); To čo je na adrese CAR sa nahrá do CBR
  - Vykonajú sa riadiace signály podľa inštrukcie v CBR,
  - Vypočíta sa ďalšia vykonávaná inštrukcia

Realizácia CLU mikropromovateľnou jednotkou(3)



Dekóder pre riadiace signály len pre vertikálny formát

Zvyšovanie výkonu procesora

- Vyššia taktovacia frekvencia
- Vylepšená architektúra
  - Pridanie nových inštrukcií
    - Napr. koprocesor pre reálnu aritmetiku
    - MMX pre multimediálne aplikácie
  - Paralelné spracovanie inštrukcií
    - Pipeling
  - Realizácia CPU hardwired
    - RISC procesory

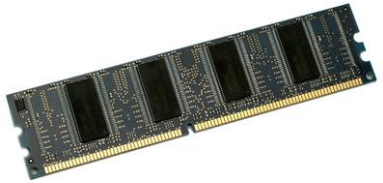
Porovnanie RISC vs. CISC

- RISC
  - Jednoduché inštrukcie vykonateľné v jednom cykle, hardvérovo realizovateľné
  - Komunikácia s pamäťou len cez load, store, menej adresných módov
  - Vysoké zreťazenie
  - Zložitosť kompilátorov, najmä pri optimalizácii kódu
- CISC
  - Zložené inštrukcie, vykonávané na viac cyklov
  - Každá inštrukcia môže komunikovať s pamäťou, mnoho adresných módov
  - Malé zreťazenie – nie všetky inštrukcie sa vykonávajú rovnako
  - Inštrukcie realizované mikroprogramom – ľahko sa realizuje kompatibilita

Výhody a nevýhody RISC

- Výhody
  - Rýchlosť
  - Jednoduchý hardware
  - Kratšia doba vývoja
- Nevýhody
  - Rýchlosť závisí od kvality kódu resp. kompilátora
  - Ladenie je komplikované pri optimalizovanom kóde
  - Zväčšovanie kódu
  - Potrebujú “drahé” CACHE pamäte najmä pre pipelining

Obsah



- Parametre pamäte
- Rozdelenie pamätí
- Triedy pamätí
- RAM
- Cache
- Stack
- Externé pamäte
  - disky, CD, DVD

Parametre pamäte

- Kapacita
  - dĺžka slova – napr. 8,16,32 bitov
  - počet slov
  - celková kapacita – veľkosť informácie, ktorú možno uchovať v pamäti
- Jednotka presunu
  - blok
  - slovo, typicky sú to bity pre reprezentáciu čísla, inštrukcie
  - adresovateľná jednotka, mnoho systémov dovoľuje adresovať na úrovni bytov

Prístup k uloženým dátam

- Ľubovoľný prístup (Random Acces) Každé slovo má pridelenú jednoznačnú adresu. Vyhľadanie slova trvá rovnaký čas. Napr. operačná pamäť.
- Sekvenčný prístup(Sequential Acces) Prístup je daný lineárnou sekvenciou. Trvanie je rozdielne, závisí od umiestnenia prvku. Napr. kazety.
- Priamy prístup(Direct Acces) Bloky majú adresy na základe umiestnia. Prístup je zabezpečený priamym prístupom k bloku a sekvenčným vyhľadávaním prvku. Napr. disky...
- Asociatívny prístup (Content Acces) Prvok sa vyhľadáva podľa určitej vlastnosti, najčastejšie podľa časti obsahu. Napr. cache

Výkon pamäte

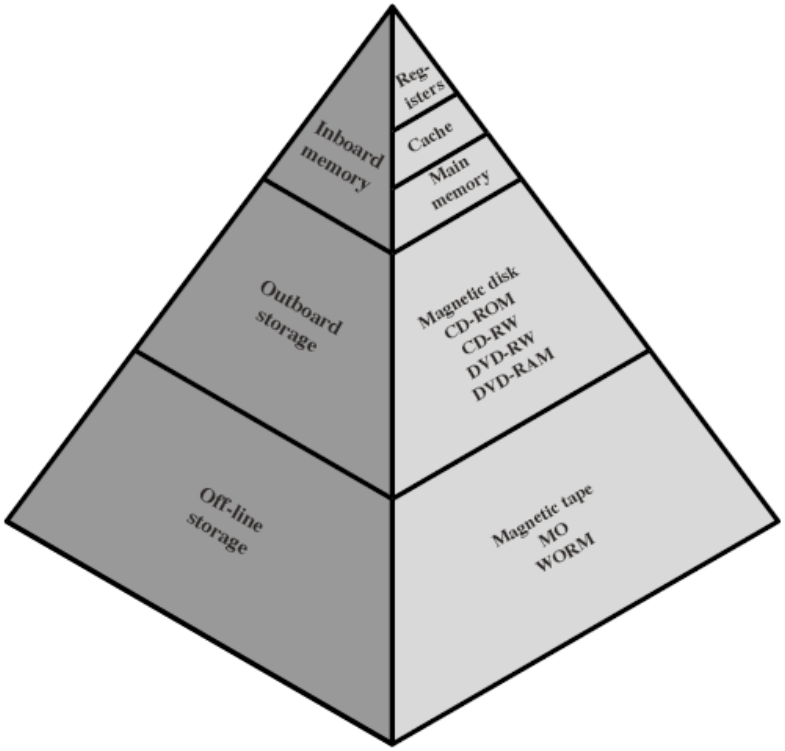
- Prístupová doba (latencia) Pre RAM je to čas R/W operácie, pre nie RAM pamäte to je čas, ktorý trvá R/W zariadeniu presunúť sa na požadovanú lokáciu.
- Čas pamäťového cyklu Len pre Ram pamäte a skladá sa z prístupovej doby a času navyše, ktorý je požadovaný pre začatie ďalšieho prístupu.
- Prenosová rýchlosť.
  - Pre RAM N/čas pamäťového cyklu
  - Pre nie RAM $T_N=T_A+N/R$
    - N počet bitov
    - R prenosová rýchlosť
    - T_A priemerná prístupová doba

Rozdelenie pamätí

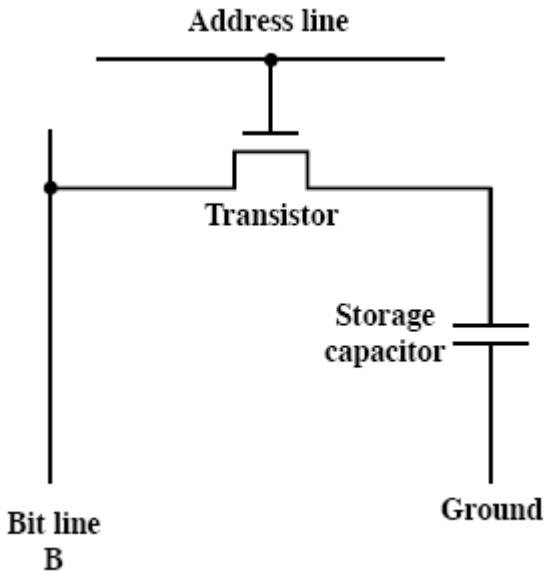
- Podľa stálosti uložených dát
  - trvalé pamäte
    - ROM programujú sa pri výrobe
    - PROM môže ich naprogramovať užívateľ, ale len raz
    - EPROM je možné prepísať viac krát ,UV svetlo
    - EEPROM je možné prepísať viac krát, elektricky
    - flash po blokoch
  - dočasné pamäte
    - dynamické DRAM, je potrebné pamäť obnovovať, v čase obnovy nie je možné pracovať s pamäťou
    - statické SRAM
- Podľa možnosti čítania a zapisovania
  - RWM read/write
    - rýchle čítanie, rýchly zápis
    - rýchle čítanie, pomalý zápis
  - ROM read only

Triedy pamätí

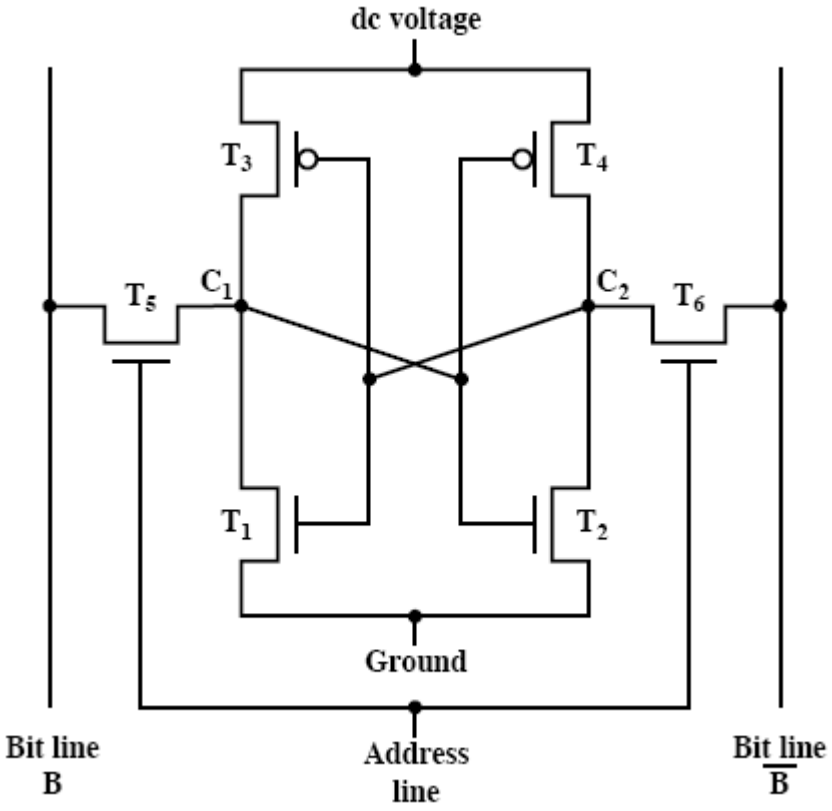
- registre procesora
- CACHE
- hlavná pamäť
- externá pamäť
- off-line externá pamäť



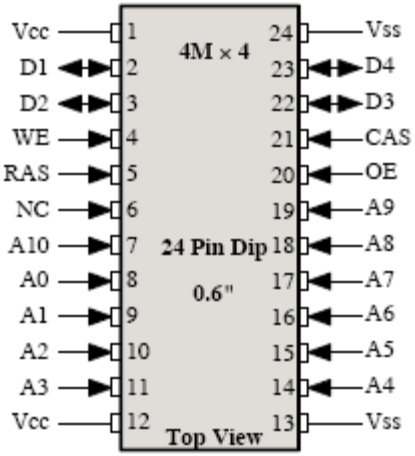
DRAM Bunka

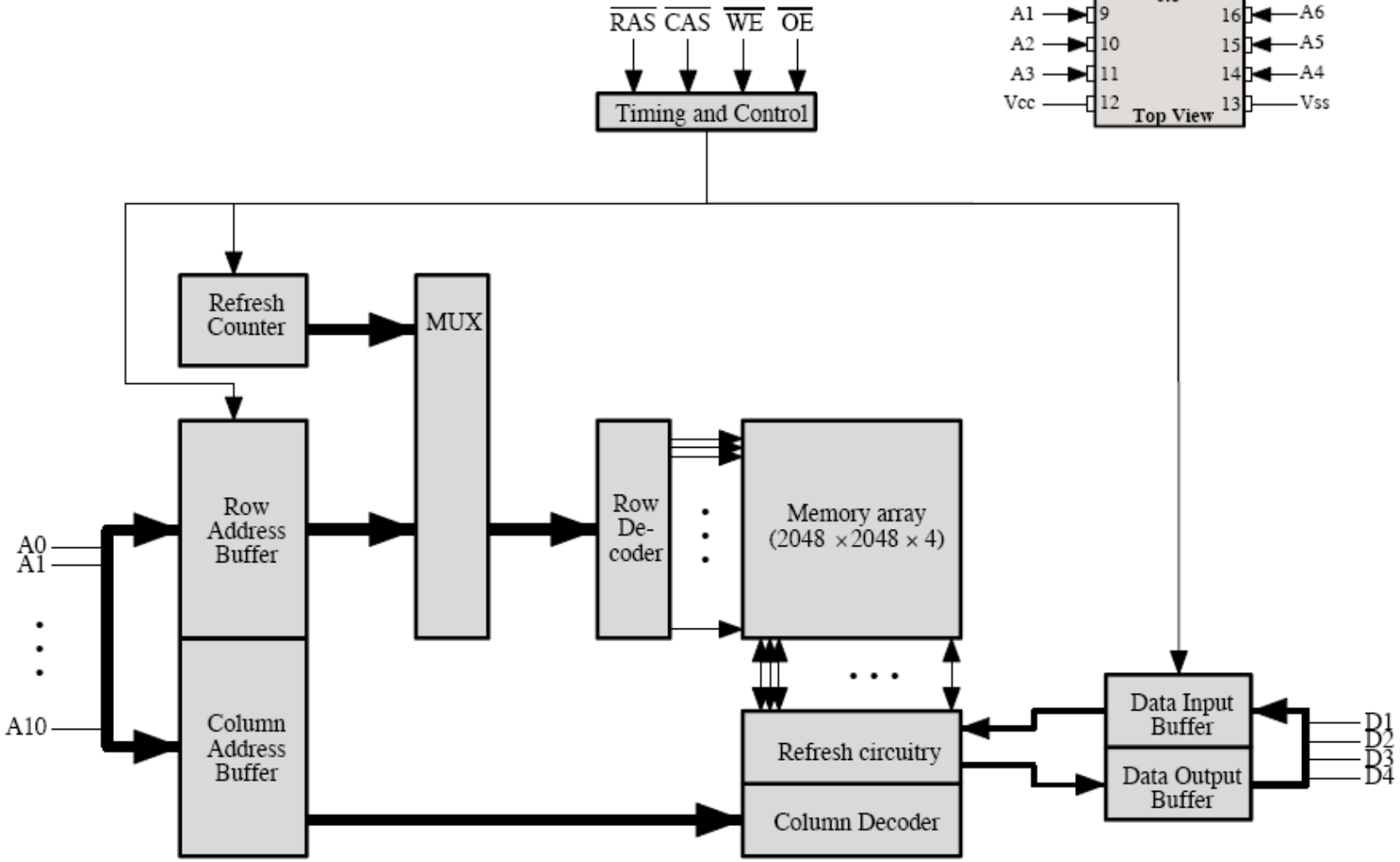

Address line
Transistor
Storage
capacitor
Bit line
B
Ground

SRAM Bunka

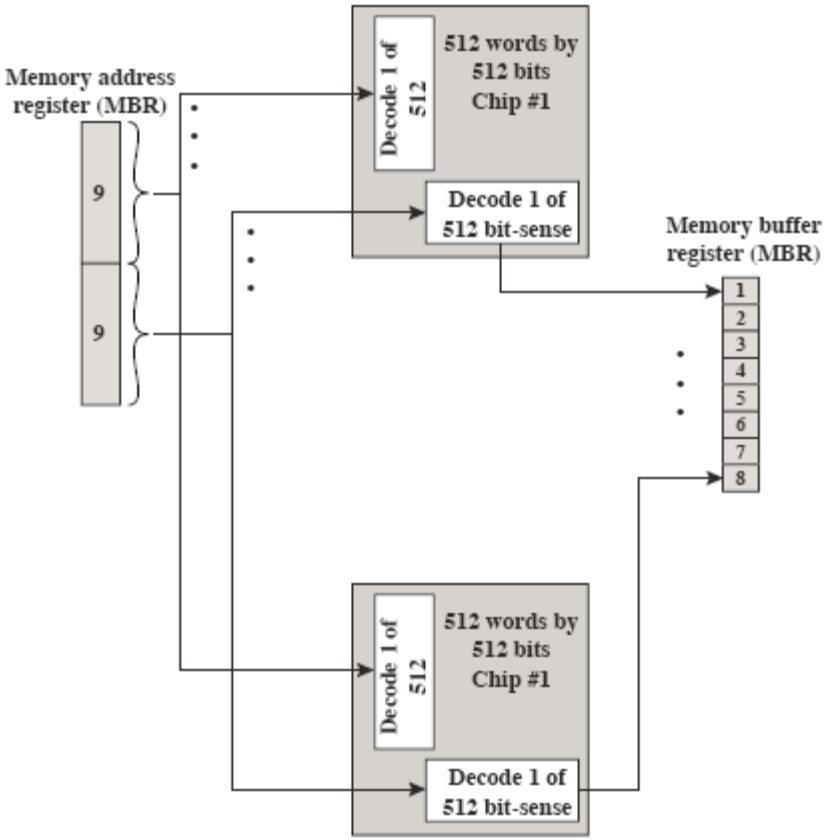

dc voltage
T3
T4
T5
C1
C2
T6
T1
T2
Ground
Bit line
B
Address
line
Bit line
B

DRAM 16 Megabit

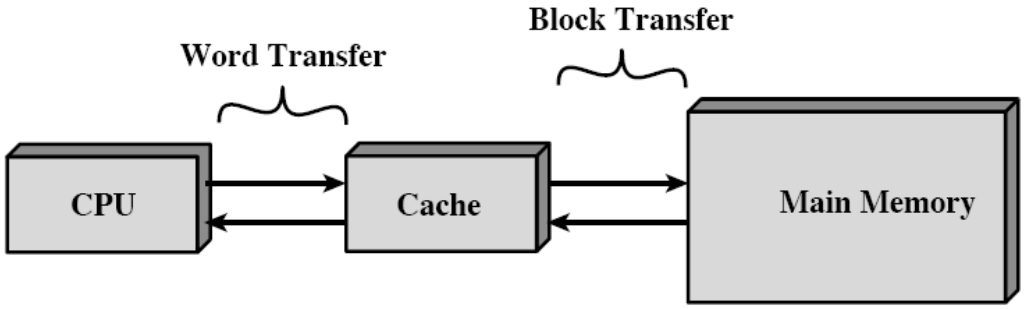


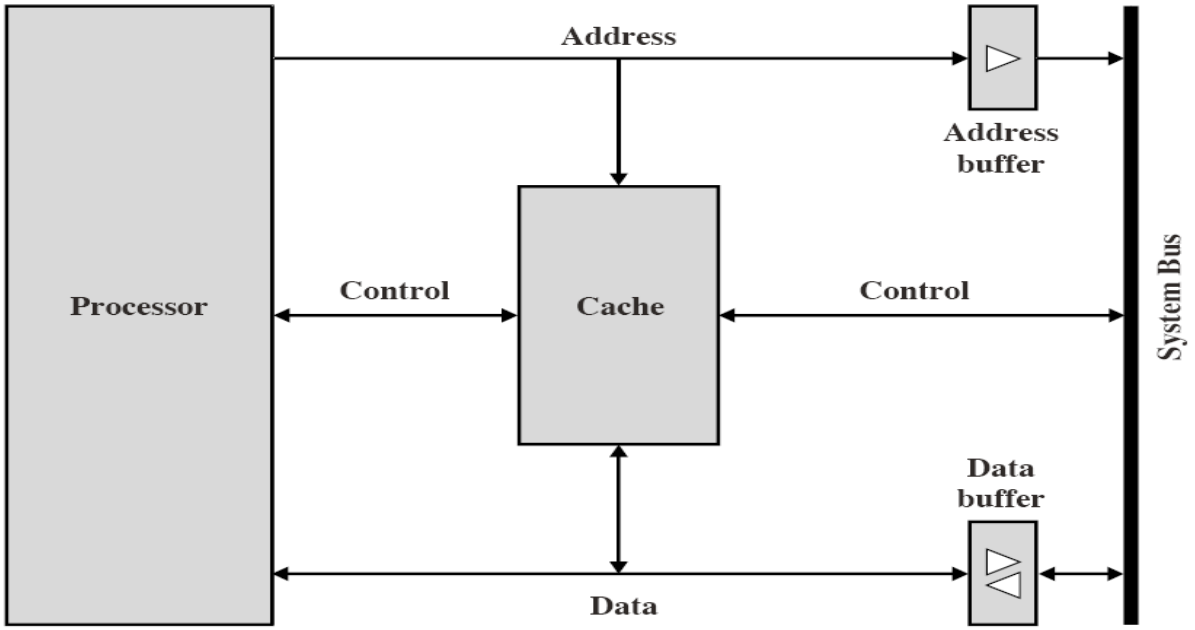


Organizácia 256 Kbyte pamäte



Cache pamäť(2)



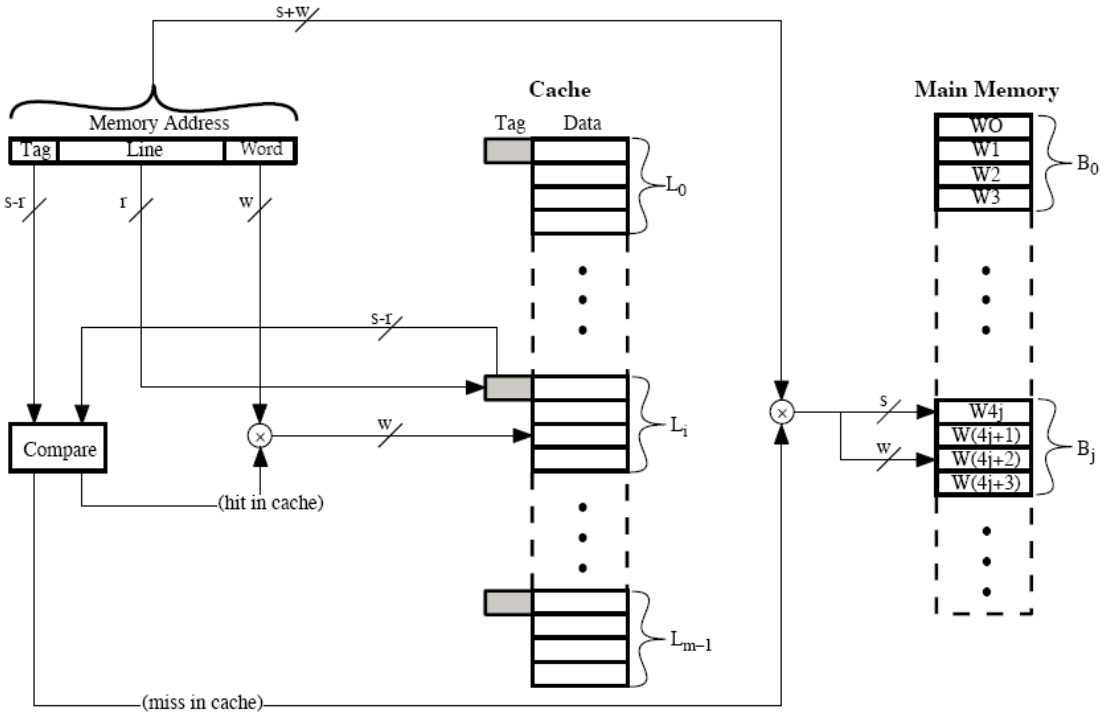


Cache pamäť

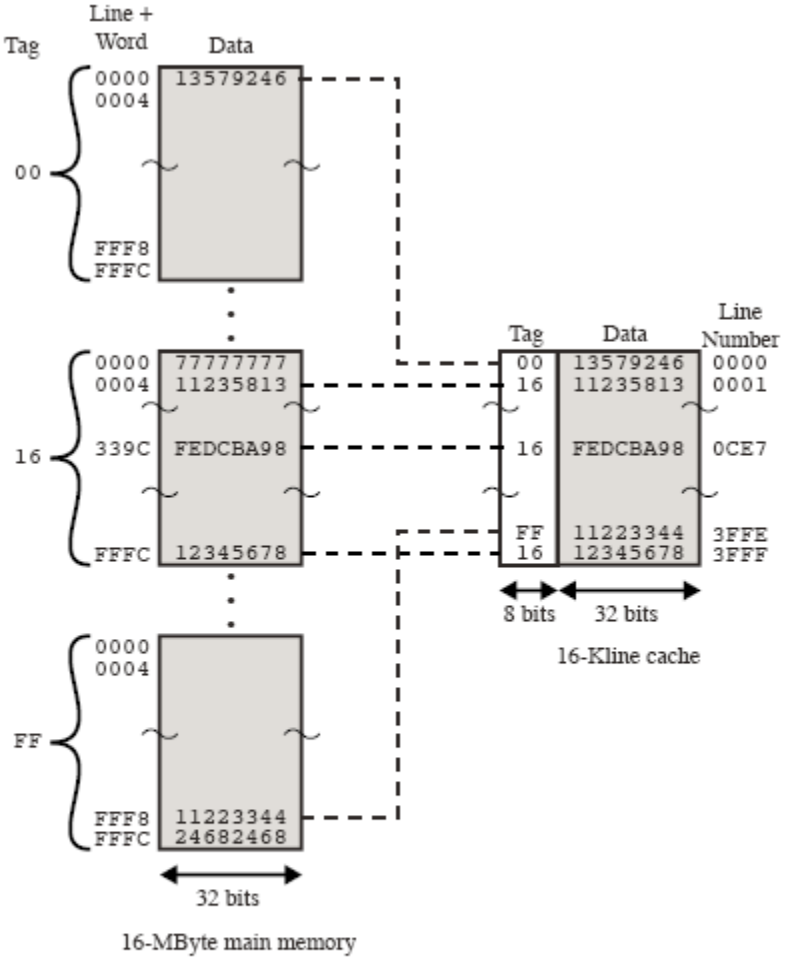
- Princíp
  - časovej lokality. Adresa, ktorá bola vyvolaná bude vyvolaná znova.
  - miestnej lokality. V krátkej dobe sa bude čítať z okolia aktuálne čítaného údaju
- nízka kapacita
- slúži na ukladanie najpotrebnejších údajov
- Prístupy
  - LFU – vylúči sa bunka, ktorá sa používa najmenej
  - LRU – vylúči sa bunka, ktorá sa nepoužívala najdlhší čas
- Zápis do CACHE
  - pri každom zápise, sa uskutoční aj zápis do hlavnej pamäte
  - ak nie je potrebné uchovávať položku, tak pred vymazaním sa obsah kópie zapíše hlavnej pamäte

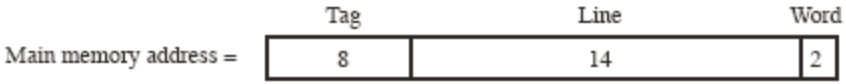
Priame mapovanie v Cache

adresa = index+tag+word, do chache sa uloží slovo+tag,
i= j mod m, j je číslo bloku v pamäti, m počet riadkov v cache



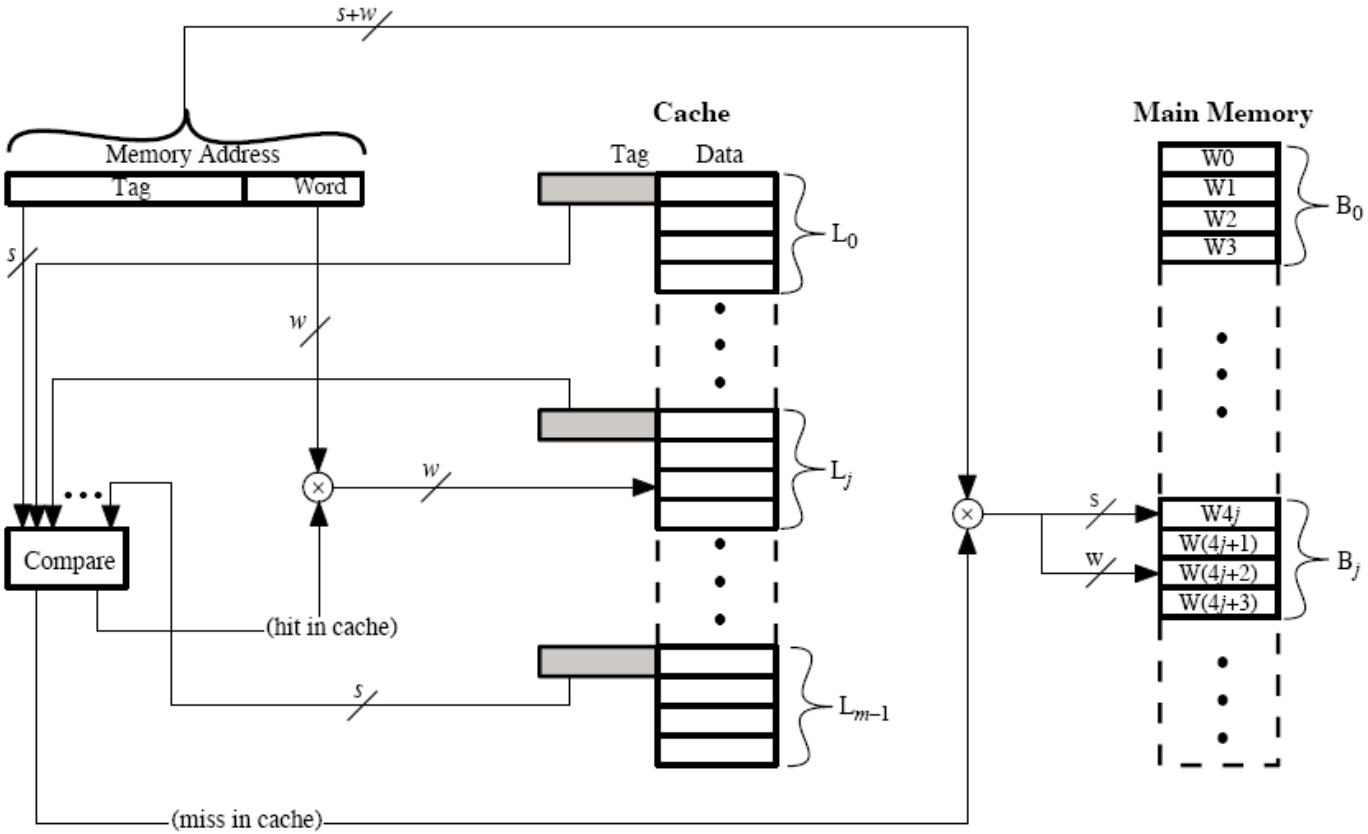
Príklad priameho mapovania





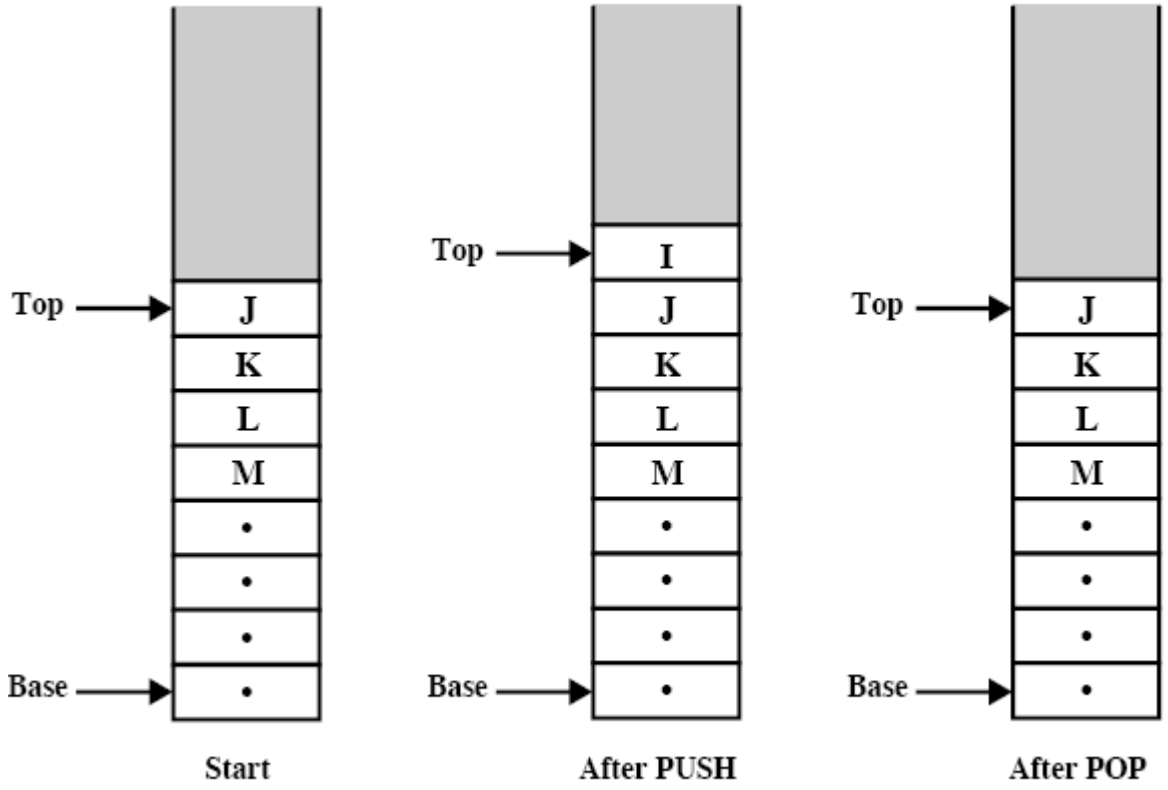
Asociatívne mapovanie

- adresa= tag+word



Zásobník-Stack

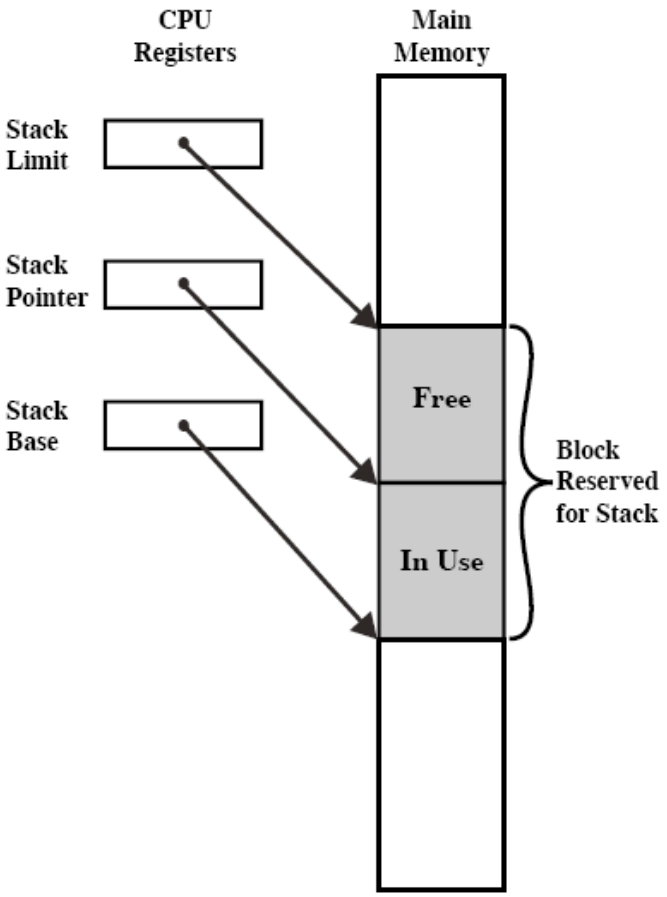
- PUSH, POP, EMPTY, FULL



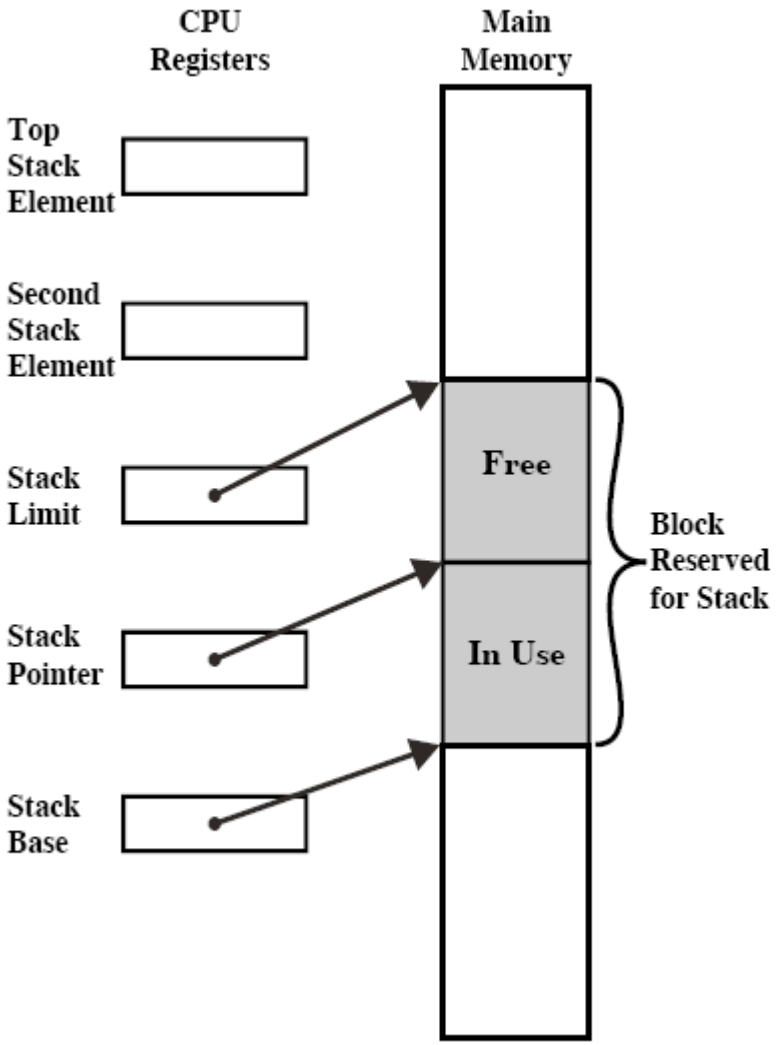
Hardvérová realizácia zásobníka

- k n-bitových registrov s paralelným zápisom a čítaním
- n posuvných k-bitových registrov. Operácie PUSH, POP sa realizujú pomocou posuvou(ako?)

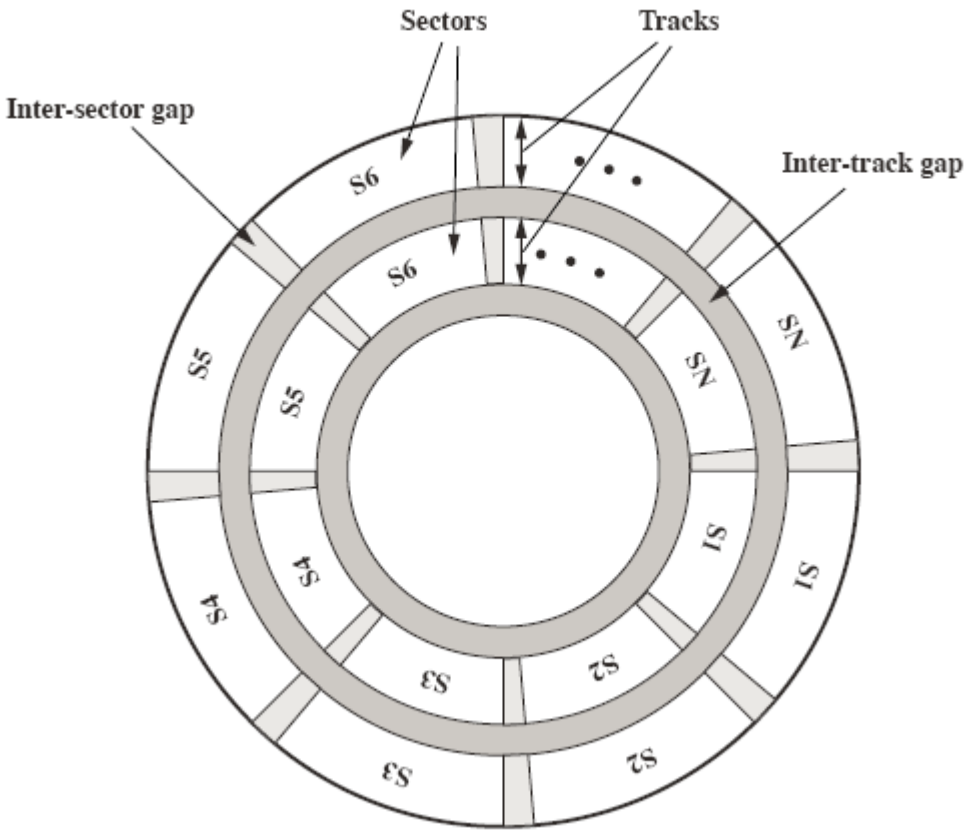
Celý zásobník v pamäti



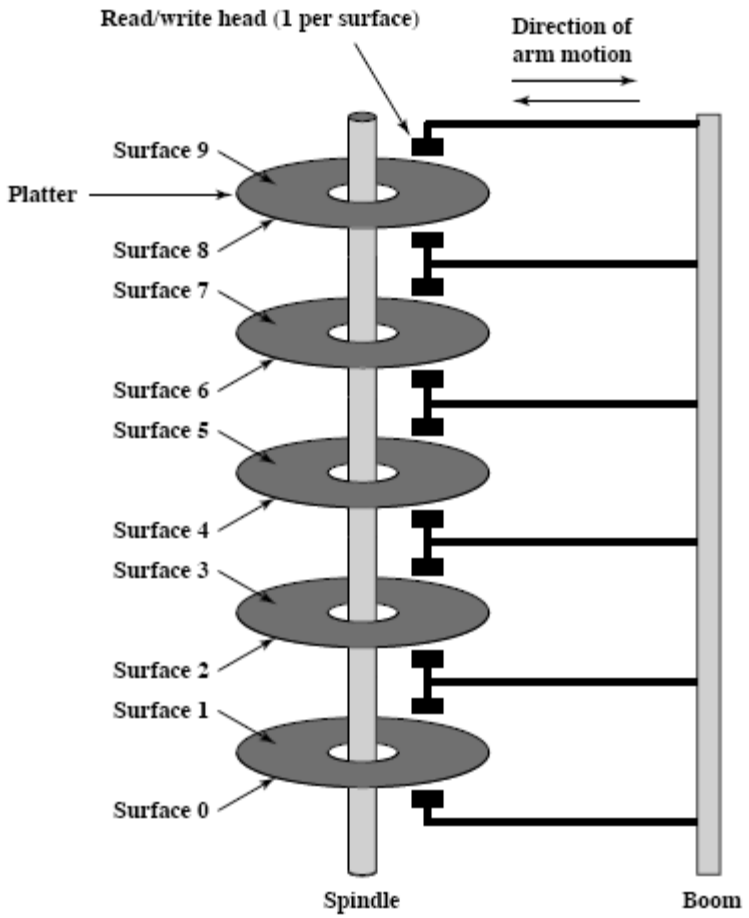
Kombinovaná realizácia



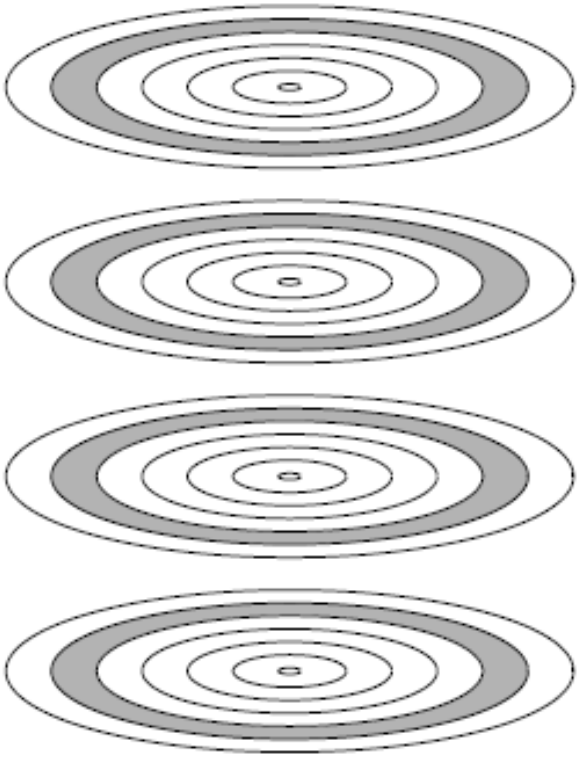
Magnetické disky



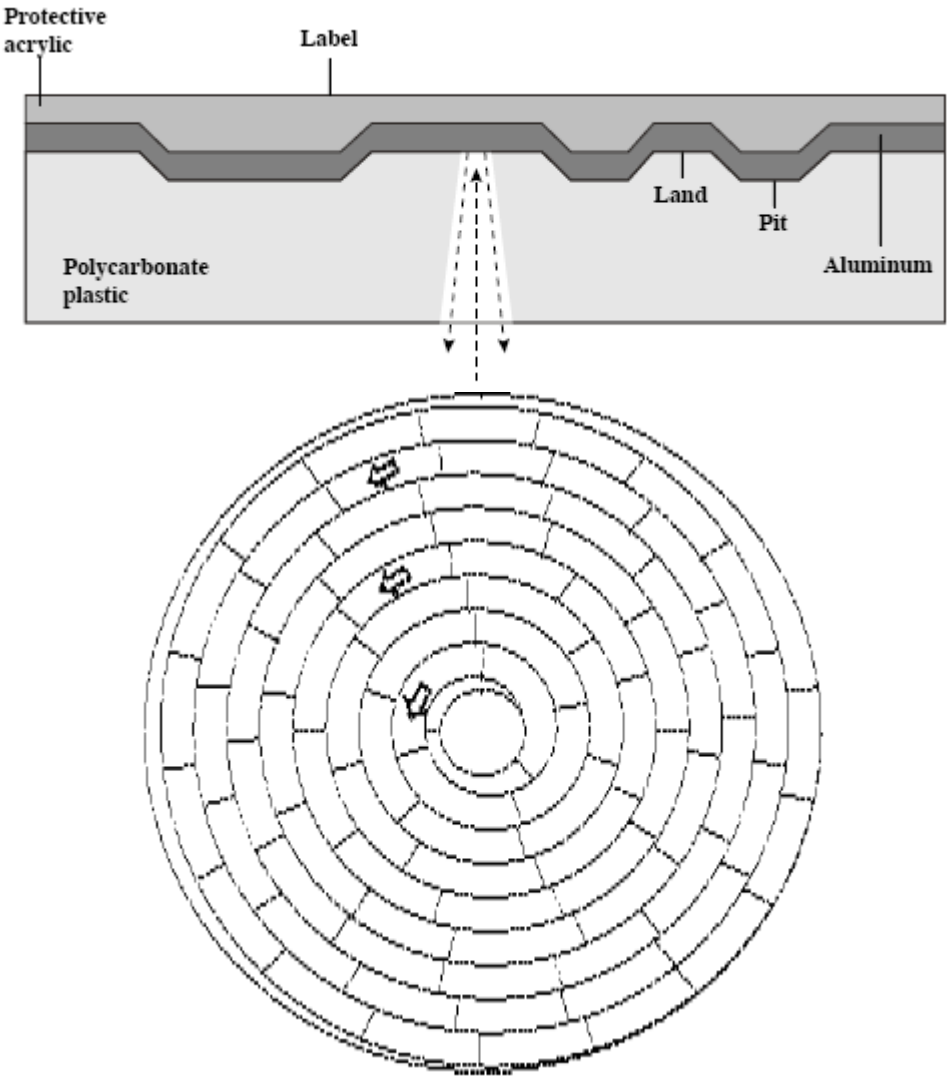
Členenie disku- plochy



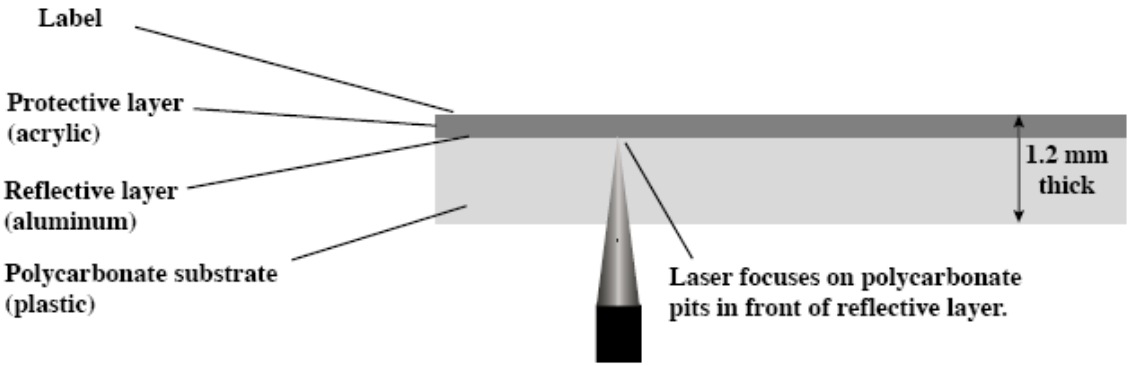
Členenie disku- cylindre



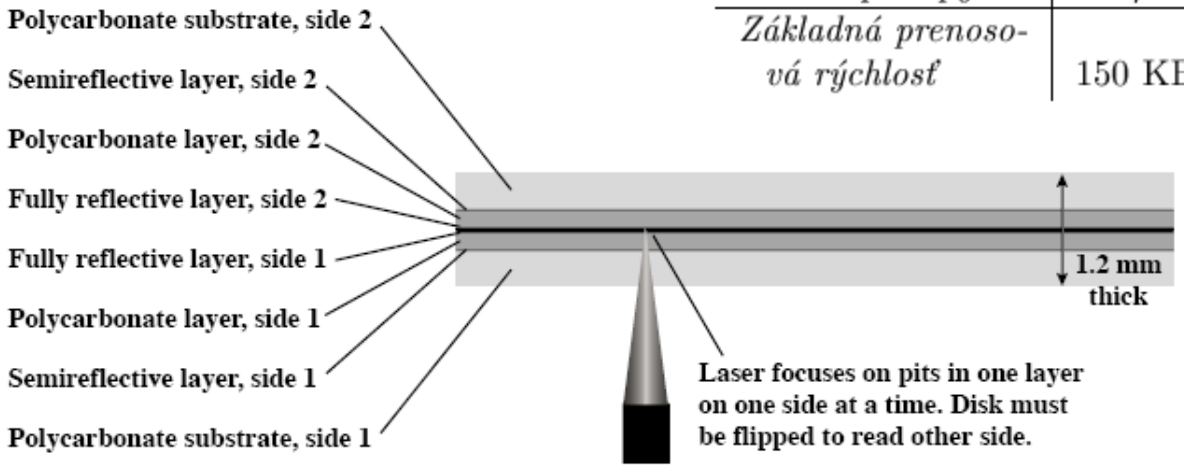
CD


Protective acrylic
Label
Land
Pit
Aluminum
Polycarbonate plastic

CD vs. DVD



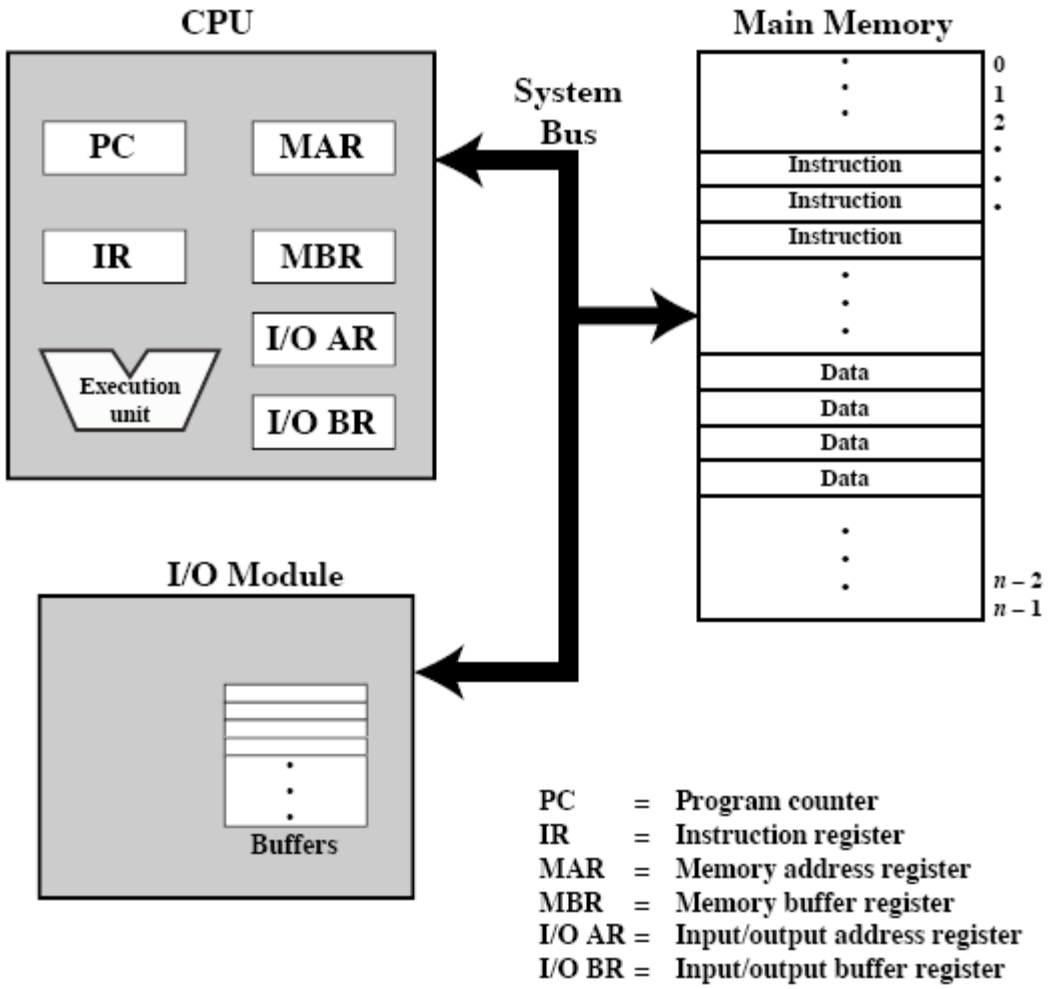
	CD-ROM	DVD
Hrúbka disku	1.2mm	0.6 mm (jednostranný) 1.2 mm (obojstranný)
Veľkosť pitu	0.83 μm	0.4 μm
Rozostup stopy	1.6 μm	0.74 μm
Základná prenosová rýchlosť	150 KB/s	11 MB/s



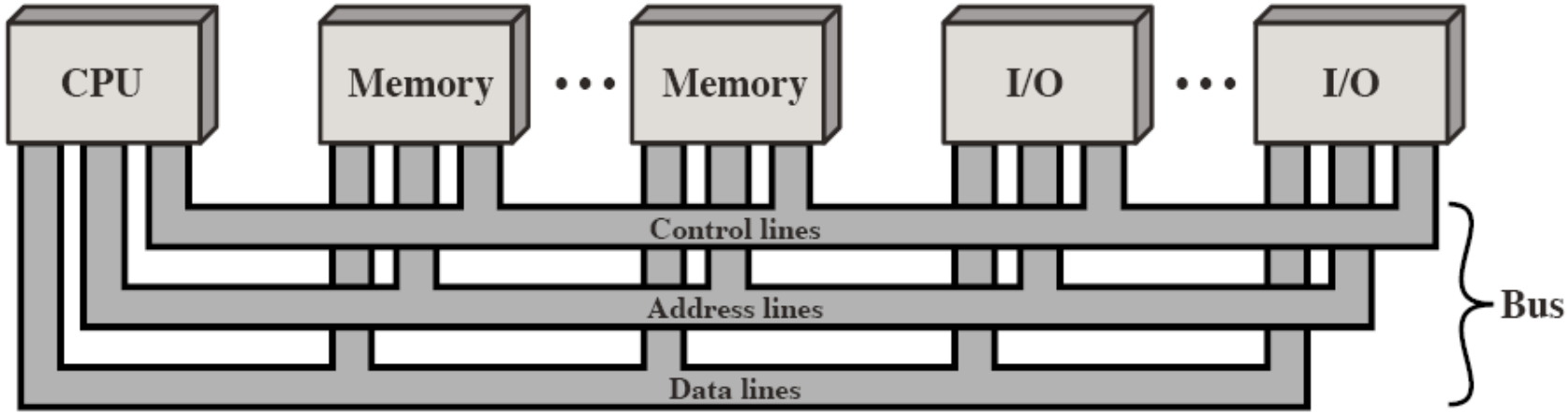
Obsah

- Zloženie I/O systému
- Prístup k I/O zariadeniam
- Prenos dát
- Programové riadenie prenosu dát
- Prerušenie
- DMA

Komponenty počítača



Komponenty počítača(2)



Zloženie I/O systému

- I/O zariadenia
  - vstupné, napr. myš
  - výstupné, napr. tlačiareň
  - vstupno/výstupné, napr. modem
- radiče I/O zariadení (device controller), pomocou nich zariadenie komunikuje s procesorom, alebo pamäťou, pomocou určitého protokolu.
- spoje, po ktorých sa prenášajú dáta
- obslužný software – Interupt handlery, drivery zariadenia

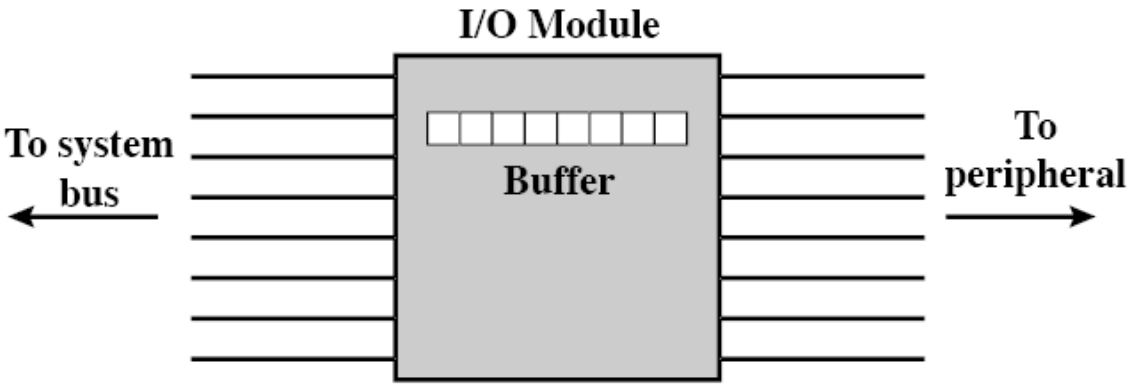
Prístup k I/O zariadeniam

- memory mapped I/O
  - I/O porty sú pripojené k adresovej zbernici, každé I/O zariadenie prekryje určité pamäťové miesta svojimi vstupmi resp. výstupmi, vstupné I/O sa správa ako ROM a výstupné ako RWM pamäť. Inštrukcia, ktorá vie pracovať s pamäťou, môže pracovať s I/O. Napr. videopamäť.
- I/O mapped I/O
  - I/O porty sú nezávislé na pamäti. CPU rozlišuje, či sa jedná o operácie s pamäťou, alebo I/O zariadením. Použijeme špeciálne inštrukcie I/O zariadenia IN, OUT. Pri prenose dát sa prenáša aj riadiaci signál, ktorý rozlišuje, či sa komunikuje s pamäťou, alebo I/O zariadením.

Prenos dát

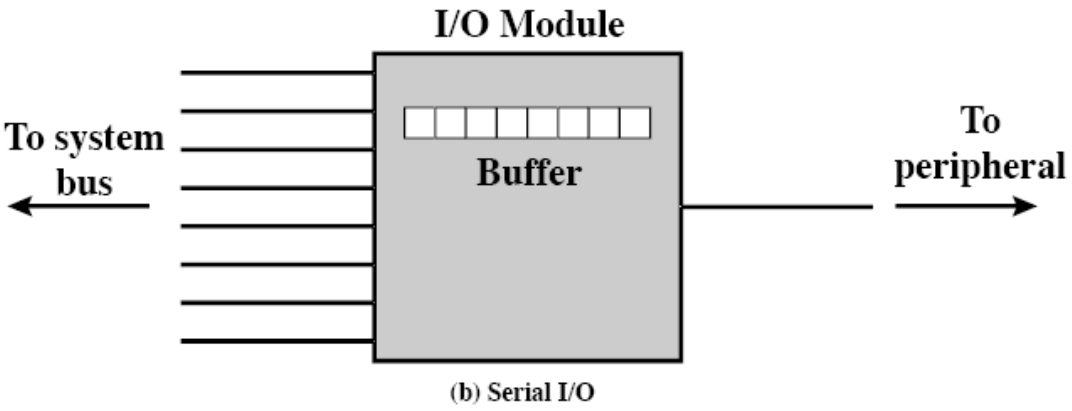
- Podľa formátu prenášaných dát rozlišujeme
  - sériový, správa sa prenáša bit po bite jednou komunikačnou linkou, je pomalší, lacnejší
  - paralelný, máme k dispozícii viac liniek, čiže možeme naraz prenášať niekoľko bitov, je rýchlejší, avšak drahší
- Podľa prenosového módu
  - synchrónny, vysielateľ aj príjemca sa dohodnú na rovnakej frekvencii vysielania a prijímania. CPU na adresovú zbernicu pošle adresu zariadenia, na dátovú dáta a nastaví signál WRITE na 1. Zariadenie prečíta dáta, pokiaľ WRITE=1. Signál WRITE je generovaný s istou frekvenciou a má rovnakú dĺžku pričom systém má
    - rôznu dĺžku synchronizačných impulzov, nejakú si dohodnú na začiatku komunikácie, podľa určitého protokolu
    - rovnakú dĺžku, prispôsobená najpomalšiemu zariadeniu
  - asynchrónny, nenastavuje sa rovnaká rýchlosť, obaja môžu vysielať rozličnými rýchlosťami. Obaja posielajú množstvo riadiacich signálov napr. request, acknowledge, potvrdenie dát a pod. Scenár komunikácie špecifikuje protokol.

Sériový a paralelný prenos



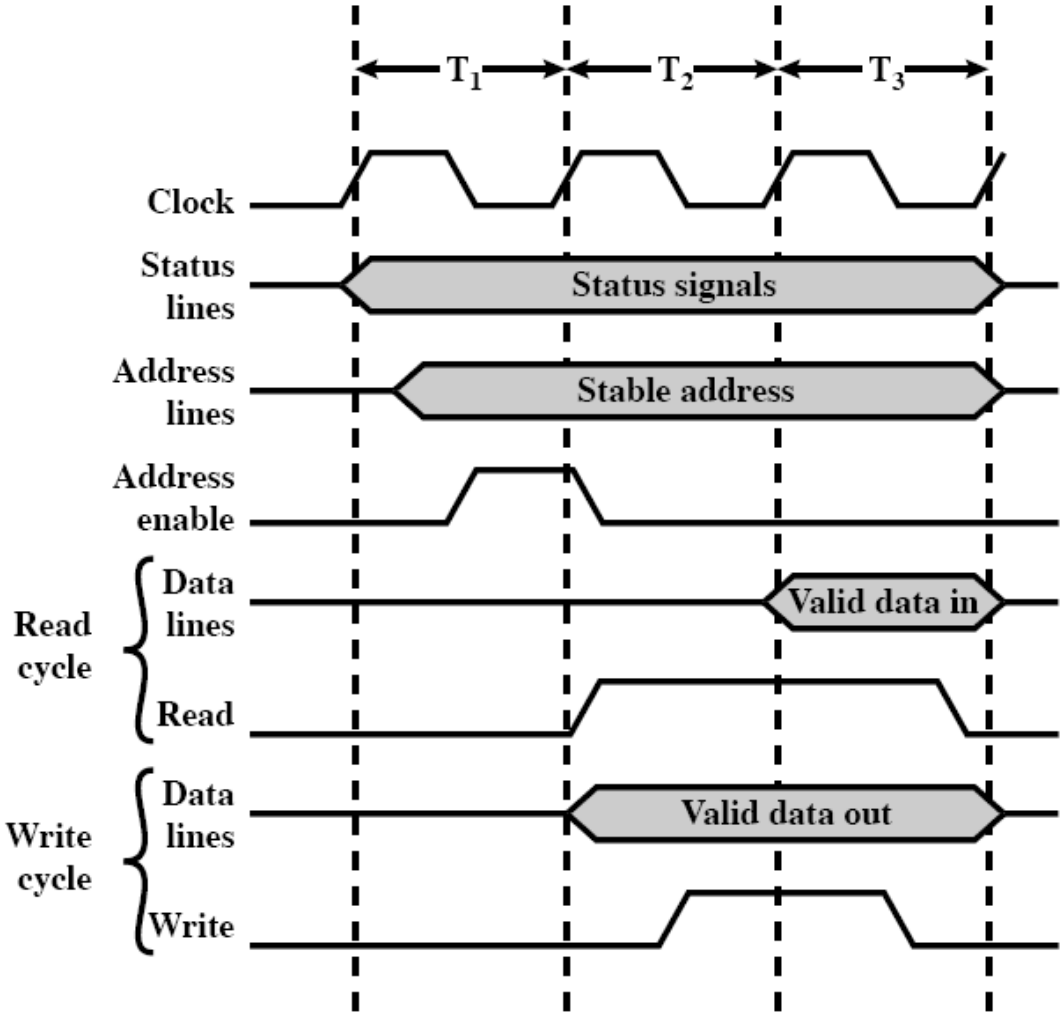
paralelný

sériový

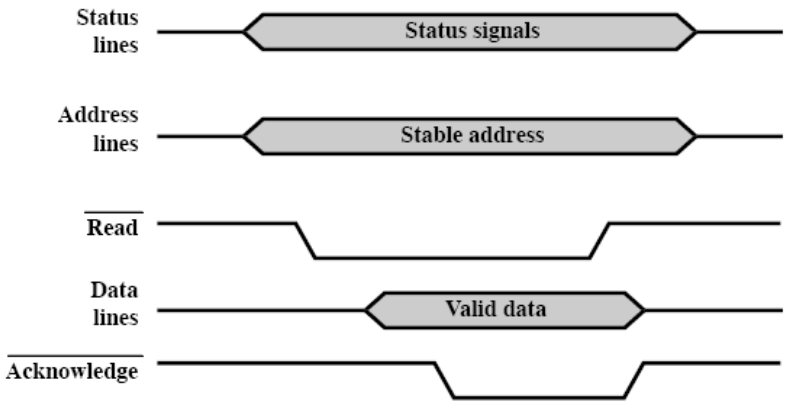


(b) Serial I/O

Časovanie synchrónneho prenosu

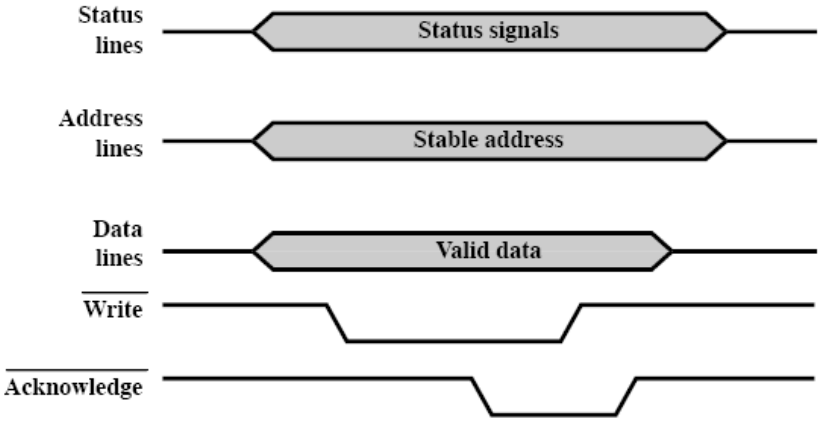


Časovanie asynchrónneho prenosu



čítanie

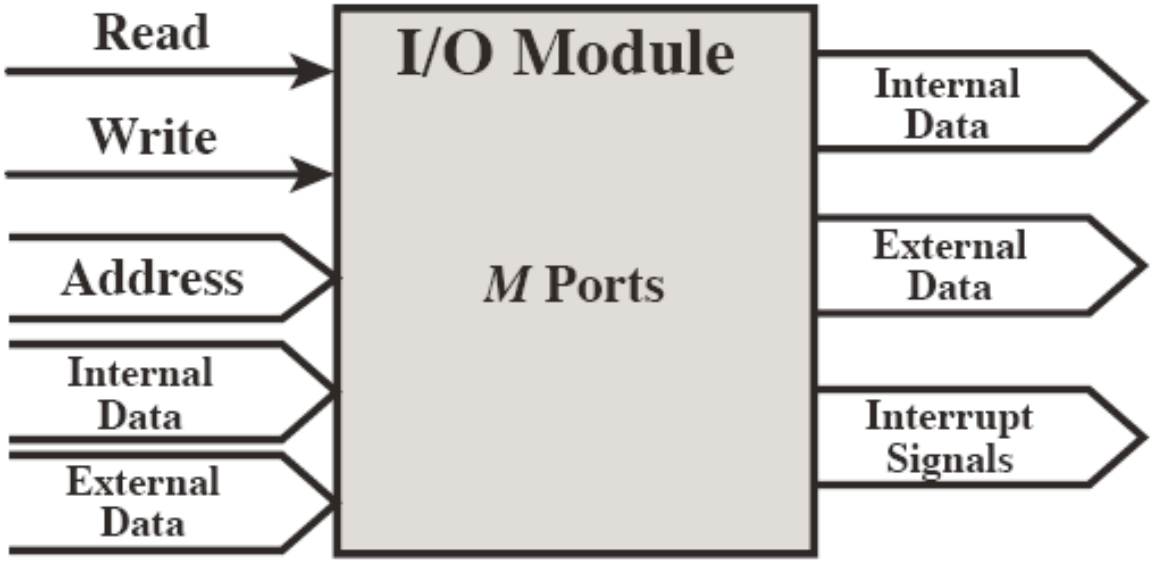
zápis



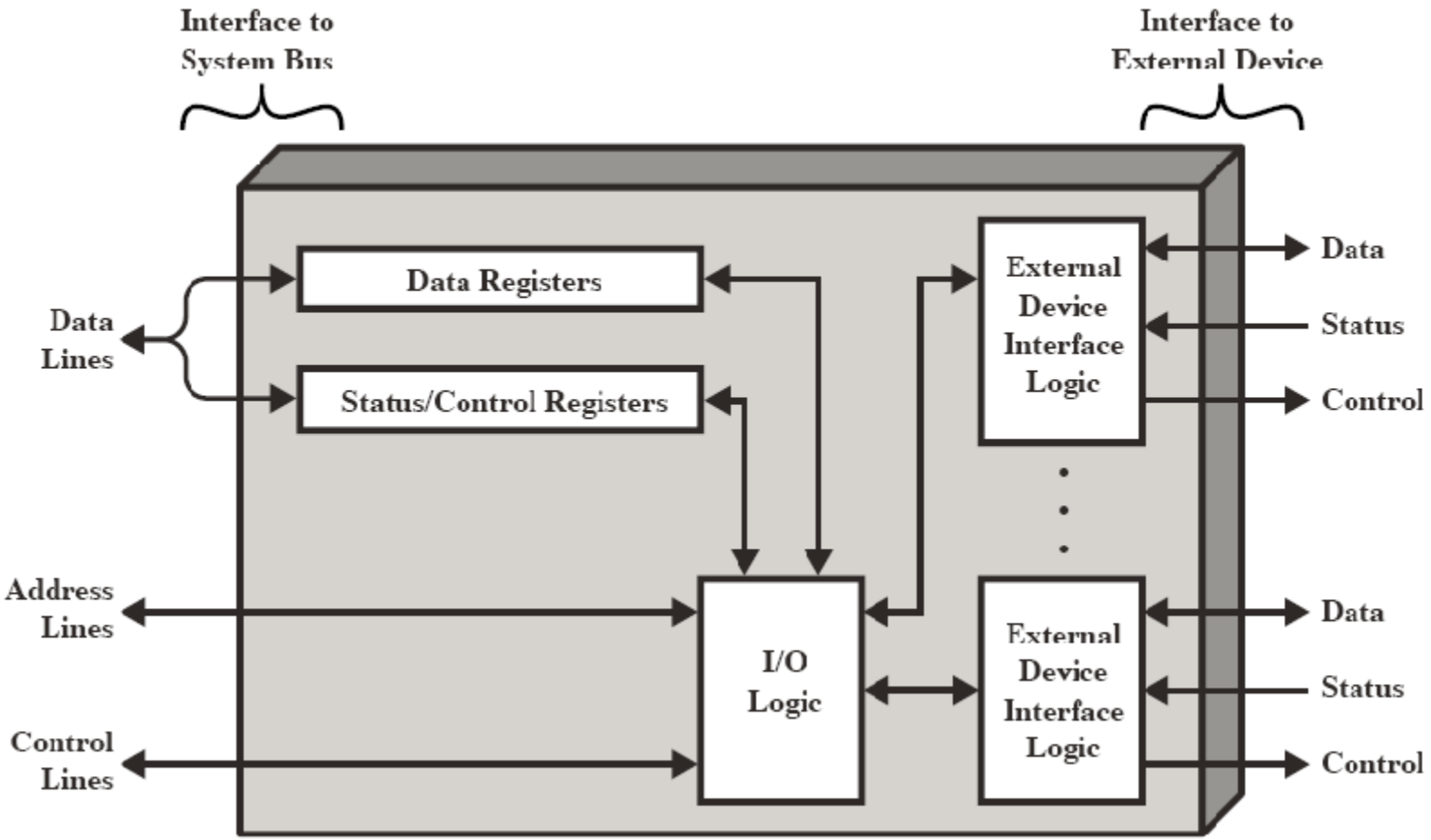
Riadenie prenosu dát

- Programom riadené I/O
  - inicializácia, prenos a ukončenie spojenia je softwérová, prenos údajov prebieha pomocou CPU
- I/O riadené pomocou prerušení
  - iniciatíva je u zariadenia, realizácia pomocou prerušení
- DMA prístup do pamäte(Direct Memory Acces)
  - slúži na prenos bloku dát bez účasti procesora

I/O modul(1)


Read
Write
Address
Internal Data
External Data
I/O Module
M Ports
Internal Data
External Data
Interrupt Signals

I/O modul(2)


Interface to System Bus
Interface to External Device
Data Registers
Status/Control Registers
Data Lines
Address Lines
Control Lines
I/O Logic
External Device Interface Logic
External Device Interface Logic
Data
Status
Control
Data
Status
Control

I/O hardware

- Registre
  - status register, obsahuje informáciu o aktuálnom stave I/O zariadenia napr. či je v synchrónnom, asynchrónnom režime, či je zariadenie pripravené, či sa má zapisovať, alebo čítať
  - buffer register slúži na dočasné uloženie, ktoré treba preniesť, resp. na dočasné uloženie prijatých údajov, pokiaľ sa nespracujú
  - data counter udáva koľko dát ešte treba preniesť
  - buffer pointer uchováva adresu pamäťového miesta, kam sa majú ukladať informácie z buffer registra

Programom riadené I/O

- Nastaví sa buffer pointer na začiatok prenášaného bloku dát
- do data counter sa zapíše veľkosť bloku
- cyklicky sa opakuje prenos slov bloku, t.j. pokiaľ data counter>0
  - CPU overí, či je zariadenie pripravené t.j. prečíta obsah status registra
  - do buffer registra sa zapíše slovo pomocou buffer pointra, potom I/O obvod spustí ich vysielanie
  - CPU zníži data counter, zvýši buffer pointer

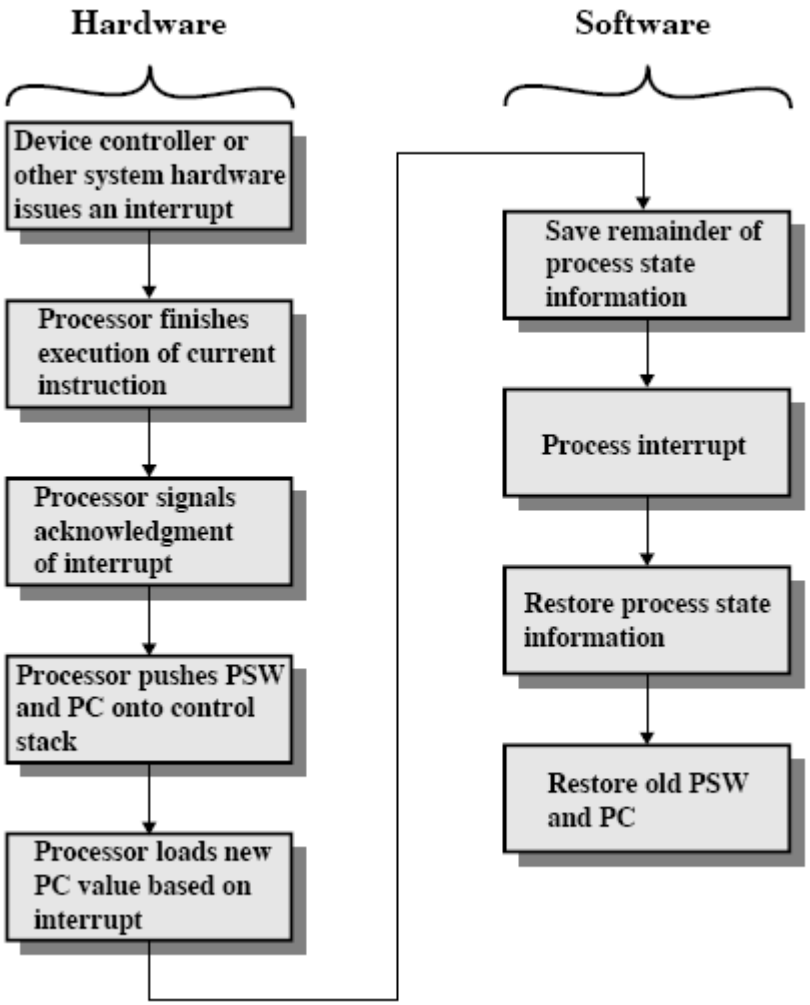
I/O riadené pomocou prerušení

- iniciatíva je u zariadenia, ktoré vygeneruje signál INTR (interupt request)
- rozlišujeme
  - maskovateľné, môžu počkať resp. ignorovať špeciálnymi inštrukciami resp. nastavením určitých riadiacich bitov
  - nemaskovateľné, nemožno zakázať, musia sa vykonať okamžite, počas vykonávania sa zakážu maskovateľné
- zariadenie majú rôznu prioritu, vyberie sa to s najvyššou

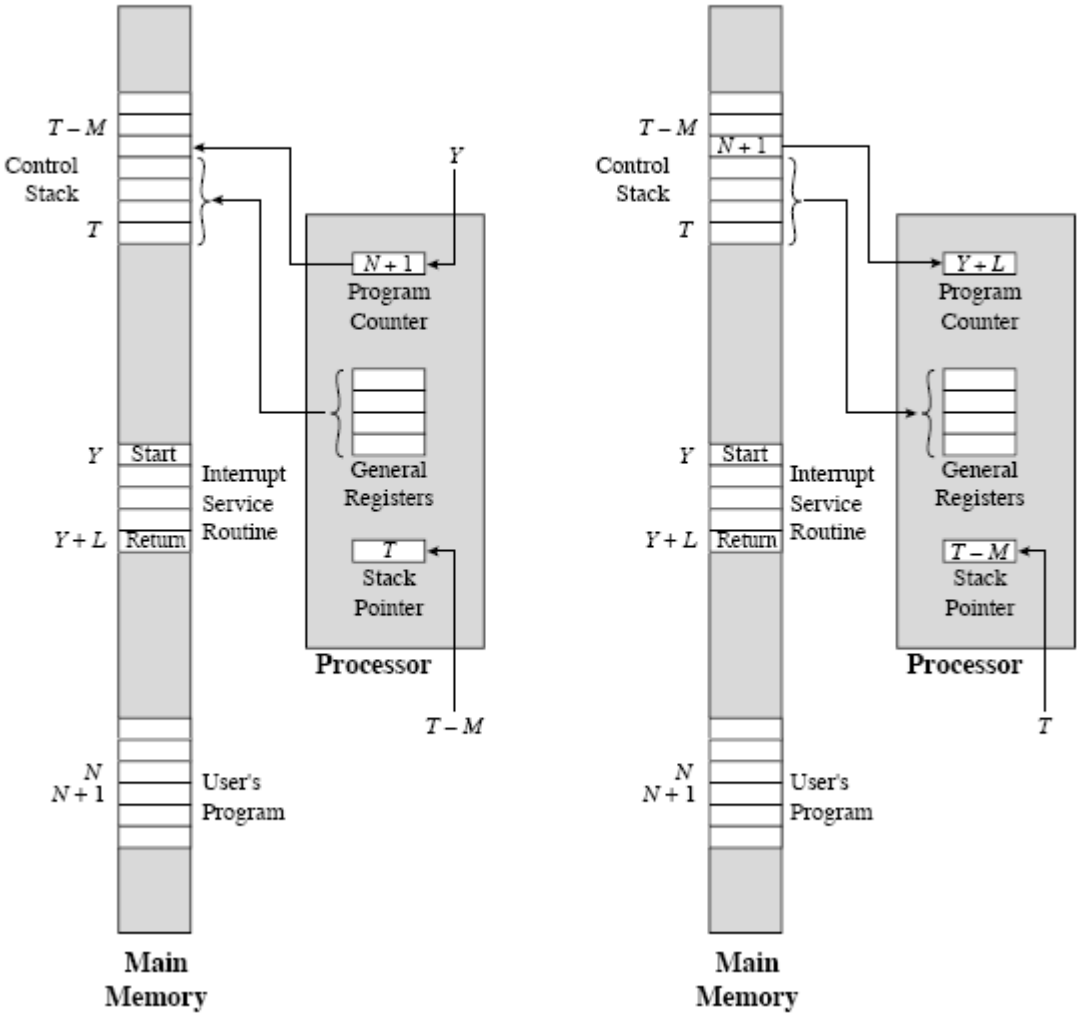
Obsluha prerušenia

- Z viac žiadostí vyberie tá s najvyššou prioritou
- Uchová aktuálny stav registrov -PSW a PC. Vloží ich do zásobníka.
- Zistí adresu obslužného programu(I/O handler), pomocou tabuľky, kde sú uložené adresy obslužných programov
- Odovzdá mu riadenie
- Po ukončení sa vráti ku pôvodnej činnosti

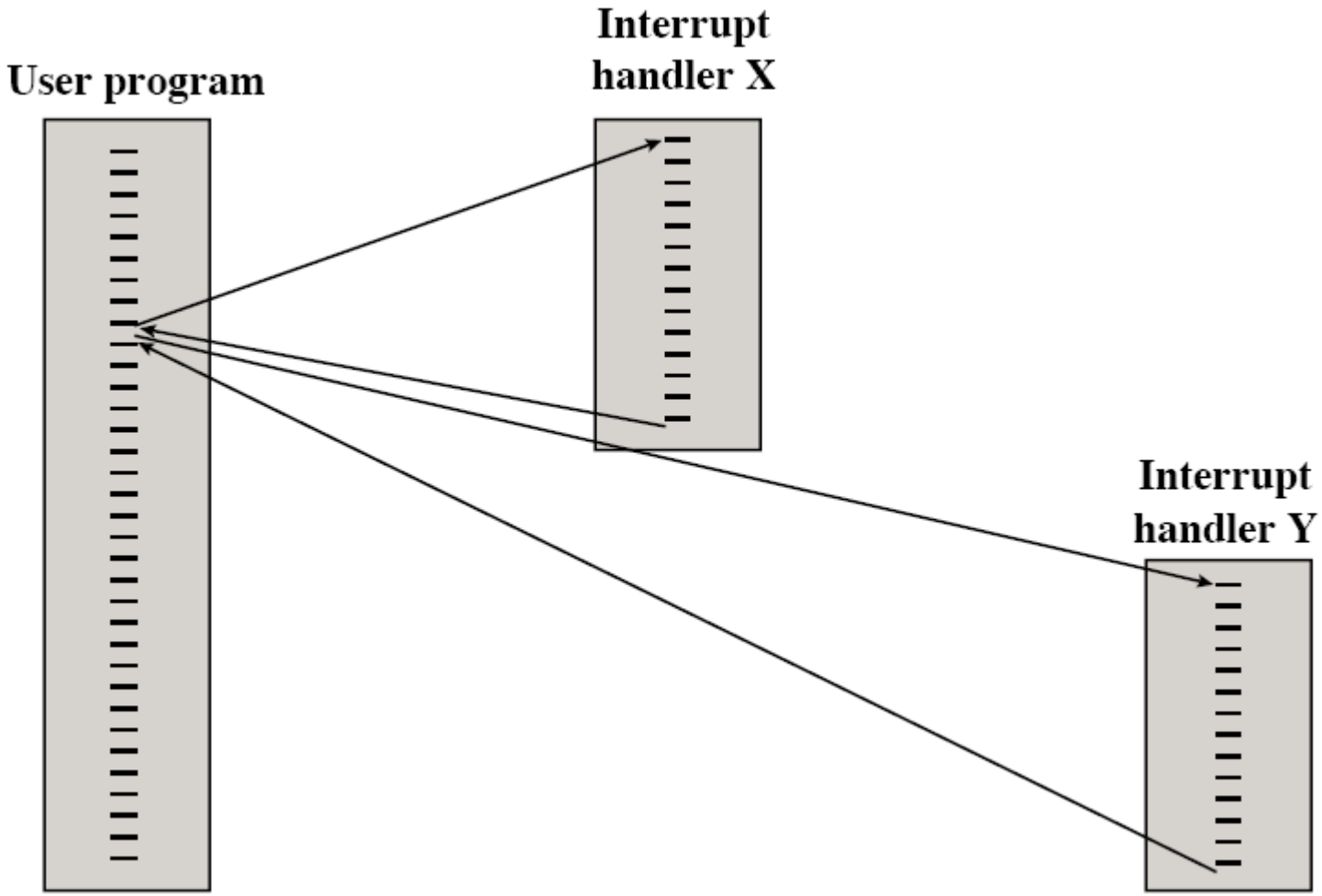
Obsluha prerušenia(2)



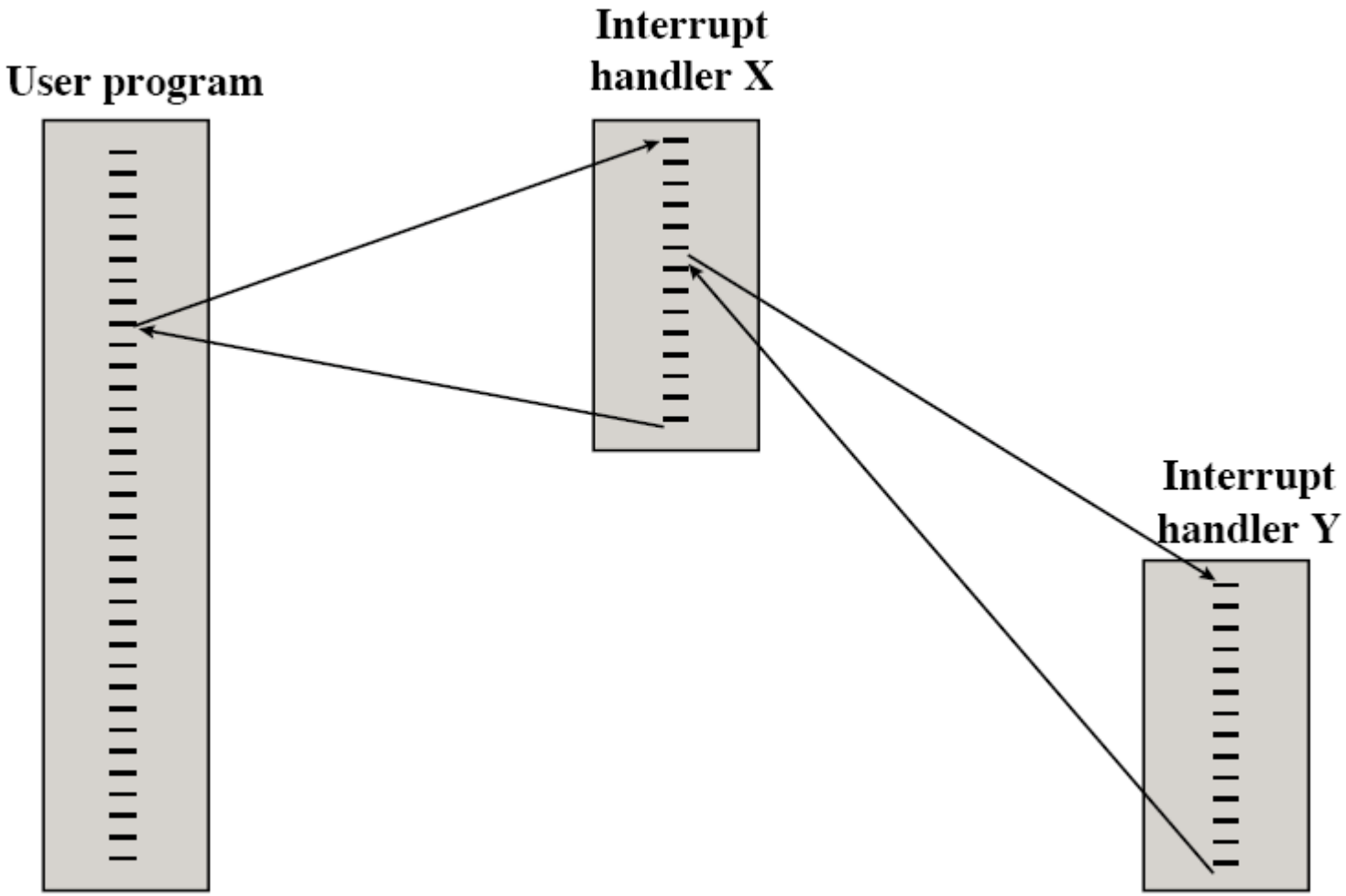
Obsluha prerušenia(3)



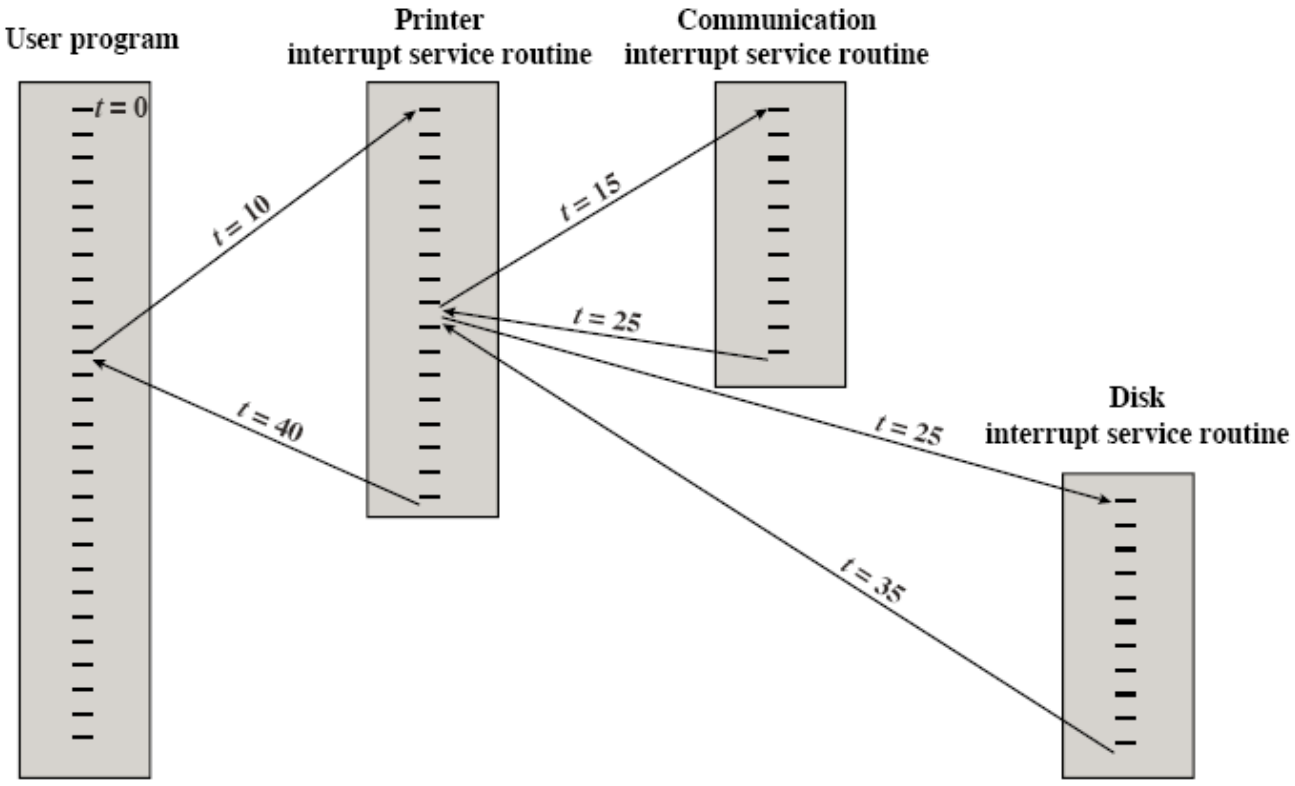
Sekvenčné vykonanie prerušení



Vnorené vykonanie prerušení



Príklad viacnásobného prerušenia (tlač dokumentu z disku na sieťovú tlačiareň)

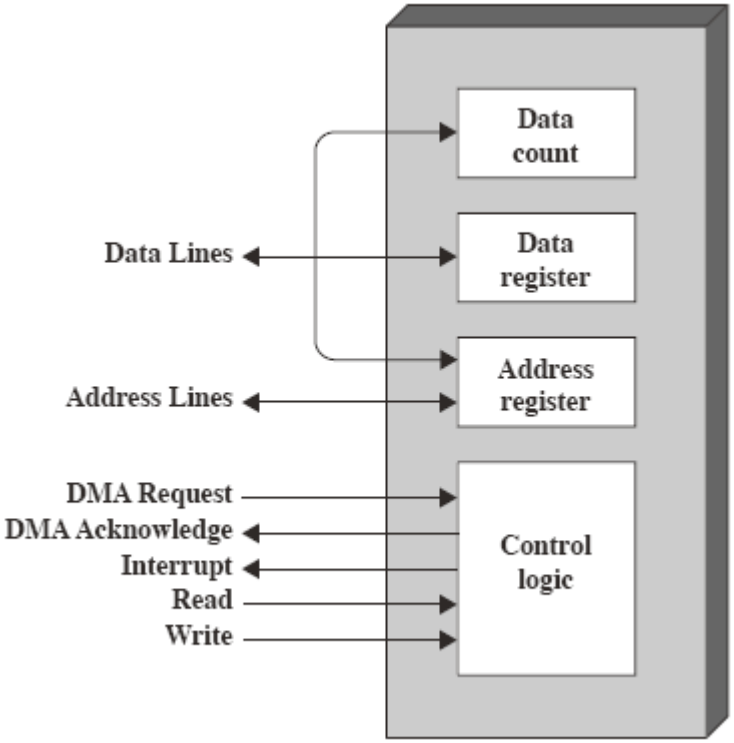


PISR má nižšiu prioritu ako CISR a DISR
CISR má vyššiu prioritu ako DISR

DMA (Direct memory acces)

- slúži na prenos bloku dát bez účasti procesora
- Riadenie programom, alebo pomocou prerušení nie je vhodné pre prenos väčších blokov dát napr. disk, CD...
- dáta sa prenášajú priamo, bez sprostredkovania procesorom
- riadenie pomocou DMA radiča

DMA radič


Data count
Data register
Address register
Control logic
Data Lines
Address Lines
DMA Request
DMA Acknowledge
Interrupt
Read
Write

DMA(2)

- Na inicializáciu sa používa INTR$_{equest}$ a INTA$_{cknowledge}$
- CPU prečíta INTR, inicializuje registre radiča a pošle signál INTA
- Radič vyšle DMAR$_{equest}$, CPU odpovie DMAA$_{cknowledge}$ a uvoľní riadenie zbernice DMA radiču, ktorý pracuje s pamäťou.

Breakpointy DMA a prerušenia