

Výkonnostní rovnice

$$T_{CPU} = IC \cdot CPI \cdot T_{CLK}$$

$\downarrow$ $\downarrow$
 počet instrukcí takty na instrukci

<http://pitbul.wz.cz>

Přesnější VR

$$T_{CPU} = \left(\sum_{i=1}^n ic_i \cdot CPI_i \right) \cdot T_{CLK}$$

$$CPI = \frac{\sum ic_i \cdot CPI_i}{IC} = \sum \frac{ic_i}{IC} \cdot CPI_i$$

pr.	$\frac{ic_i}{IC}$	CPI_i	$CPI_i \cdot \frac{ic_i}{IC}$	% z času vykonávání programu
shoky	0,15	2	0,3	11%
store	0,15	3	0,45	16%
load	0,32	5	1,6	59%
ALU	0,38	1	0,38	14%
			<u>CPI = 2,73</u>	

~~CPI~~ ~~VR~~ $T_{CPU} = 446599548 \cdot 2,73 \cdot \frac{1}{410^8} = 3,04 \text{ s}$

2x rychlejší shoky

shoky	0,15	1	0,15
store	0,15	3	0,45
load	0,32	5	1,6
ALU	0,38	1	<u>0,38</u>
			2,58

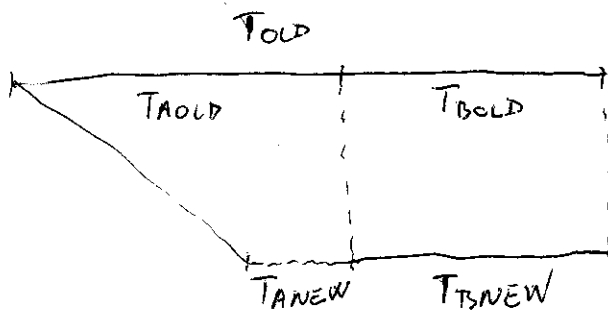
$$S = \frac{2,73}{2,58} = 1,06$$

zrychlení

2,5 krát rychlejší load

$$S = \frac{2,73}{1,77} = 1,54$$

Amotahlov zákon



$$T_{ANEW} = \frac{TAOLD}{S_E}$$

$$T_{NEW} = T_{ANEW} + T_{BSNEW}$$

$$\frac{TAOLD}{S_E}$$

$$TOLD - TAOLD$$

$$T_{NEW} = TOLD \left(\frac{TAOLD}{S_E} \cdot \frac{1}{TOLD} + 1 - \frac{TAOLD}{TOLD} \right)$$

$$\frac{1}{S_{overall}} = \frac{T_{NEW}}{TOLD} = \frac{F_E}{S_E} + (1 - F_E)$$

Pr. krytlení šoků 2x

$$S_E = 2$$

$$F_E = 11\%$$

$$S_{overall} = \frac{1}{(1 - F_E) + \frac{F_E}{S_E}} = \frac{1}{(1 - 0,11) + \frac{0,11}{2}} = 1,06$$

krytlení loadi 2,5x

$$S_E = 2,5$$

$$F_E = 59\%$$

$$S_{overall} = \frac{1}{(1 - 0,59) + \frac{0,59}{2,5}} = 1,55$$

Další příklady 1

P4 - SSE 2

- 1) 70% aplikace lze 10x rychleji pomocí SSE 2

$$S_{\text{overall}} = \frac{1}{(1-0,7) + \frac{0,7}{10}} = 2,7$$

jakou část doby výpočtu urychlí, aby celkově rychleji 2x?

$$2 = \frac{1}{(1-F_E) + \frac{F_E}{10}}$$

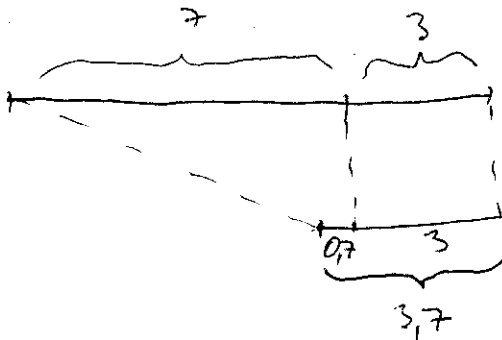
$$1) (1-F_E) + \frac{F_E}{10} = \frac{1}{2}$$

~~At~~ F_E

$$-9F_E = 5$$

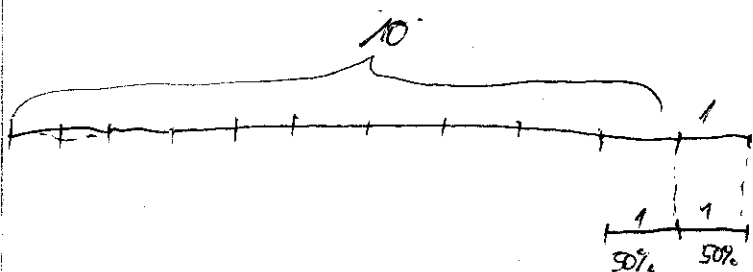
$$F_E = \frac{5}{9} = 0,55$$

1)



$$S = \frac{10}{3,7} = 2,7$$

Další příklady 2



ke princi

$$S_0 = \frac{11}{2} = 5,5$$

$$F_E = \frac{10}{11}$$

$$1 - F_E = \frac{F_E}{S_E}$$

$$S_E - S_E F_E = F_E$$

$$S_E = F_E (1 + S_E)$$

$$F_E = \frac{S_E}{1 + S_E}$$

Proč nepoužít MIPS ?

$$T_{CPU} = CPI \cdot IC \cdot T_{CLK}$$

$$b) 1,08 T_{CLK} = 10 \cdot 1955789 \cdot T_{CLK}$$

$$T_{CLK} = \frac{1,08}{1955789} = 55 \text{ ns}$$

$$a) IC = \frac{T_{CPU}}{CPI_a \cdot T_{CLK}} = \frac{13,6}{6 \cdot 55 \cdot 10^{-9}} = 41,21 \cdot 10^6$$

$$\frac{IC_a}{\#instr} = \frac{41,21 \cdot 10^6}{1955789} \approx 21 \quad \text{instr / Foperaci}$$

$$MIPS = \frac{IC}{10^6 \cdot T_{CPU}}$$

$$= \frac{IC}{10^6 \cdot CPI \cdot T_{CLK}} = \frac{1}{10^6 \cdot CPI \cdot T_{CLK}}$$

$$MIPS_a = \frac{1}{10^6 \cdot 6 \cdot 55 \cdot 10^{-9}} = 3$$

$$MIPS_b = \frac{1}{10^6 \cdot 10 \cdot 55 \cdot 10^{-9}} = 1,8$$

jak porovnávat výkonost počítačů

Počítač	C1	C2	C3
$T_{CPU}(P1)$ [s]	1	10	20
$T_{CPU}(P2)$ [s]	1000	100	20
1) celk. čas	1001	110	40
2) ar. průměr	500,5	55	20

3) váz. ar. průměr s normalizací vůči 1 poč.	hC1	1,999	10,09	20
	hC2	91,9	18,19	20

→ závisí na ref. poč.,
den vždy nejrychlejší

4) ar. průměr
zrychlení

	C1	C2	C3	$S_{C1/C2}$	$S_{C1/C3}$
1	1	10	20	0,1	0,05
2	1000	100	20	10	50
4) ar. průměr →				5,05	25,025
5) geom. průměr				1	1,58
$= \sqrt{v_1 \cdot v_2 \cdot v_3 \dots}$					

↳ normalizace
na ref. poč.

$$T = \sum w_i T_i$$

$$w_i = \frac{1}{T_i \cdot \sum \frac{1}{T_i}}$$

$$w_1(C1) = \frac{1}{1 \left(\frac{1}{1000} + \frac{1}{1} \right)} = 0,999$$

$$w_2(C2) = \frac{1}{1000 \left(\frac{1}{1000} + \frac{1}{1} \right)} = 0,001$$

$$T_{C1} = 0,999 \cdot 1 + 0,001 \cdot 1000 = 1,999$$

$$T_{C2} = 0,999 \cdot 10 + 0,001 \cdot 100 = 10,09$$

$$T_{C3} = 0,999 \cdot 20 + 0,001 \cdot 20 = 20$$

$$w_1(C2) = \frac{1}{10 \left(\frac{1}{10} + \frac{1}{100} \right)} = 0,909$$

$$w_2(C2) = \frac{1}{100 \left(\frac{1}{10} + \frac{1}{100} \right)} = 0,0909$$

$$T_{C1} = 0,909 \cdot 1 + 0,0909 \cdot 1000 = 91,9$$

$$T_{C2} = 0,909 \cdot 10 + 0,0909 \cdot 100 = 18,19$$

$$T_{C3} = \dots = 20$$

do runder do

XOR1

$$27 \leq 0 \oplus 4$$

ADDU 1

R 3E 1024.0 (400h)

OR1 R2, R0, 10

$$R_2 \leq 10$$

ADDV1

$R \leq 2048$ (POH)

LW $R_1, (R_3)$

PAK ABCD 1234

ANDI R4, R1, 0x00F

LW	IF	ID	EX	MEM	WB
ANDI	IF	ID	stall	stall	EX MEM WB

při sestavení jednolod stupně nastavil i všechny předchozí!

i 1'

"0000"

'0' when ~~pos~~ else '1' ;

napr. $Ex_{RA} = MEM.RW$

-- komentar

Cache

Pr. 1

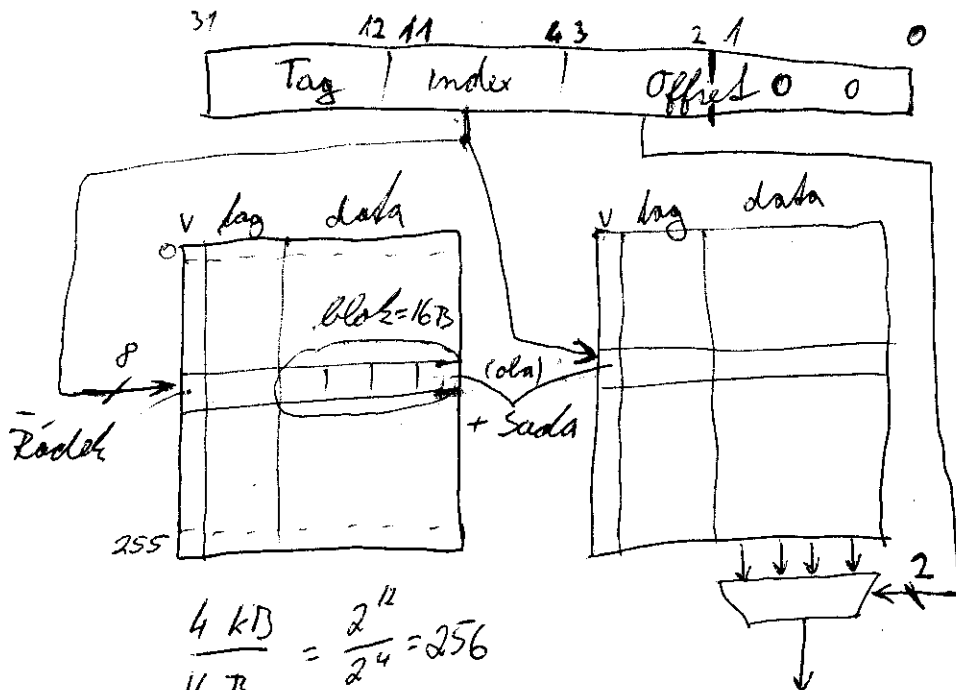
velikost 8 B (bez adresní log)

st. asoci. $S=2$

vel. bloku 16 B

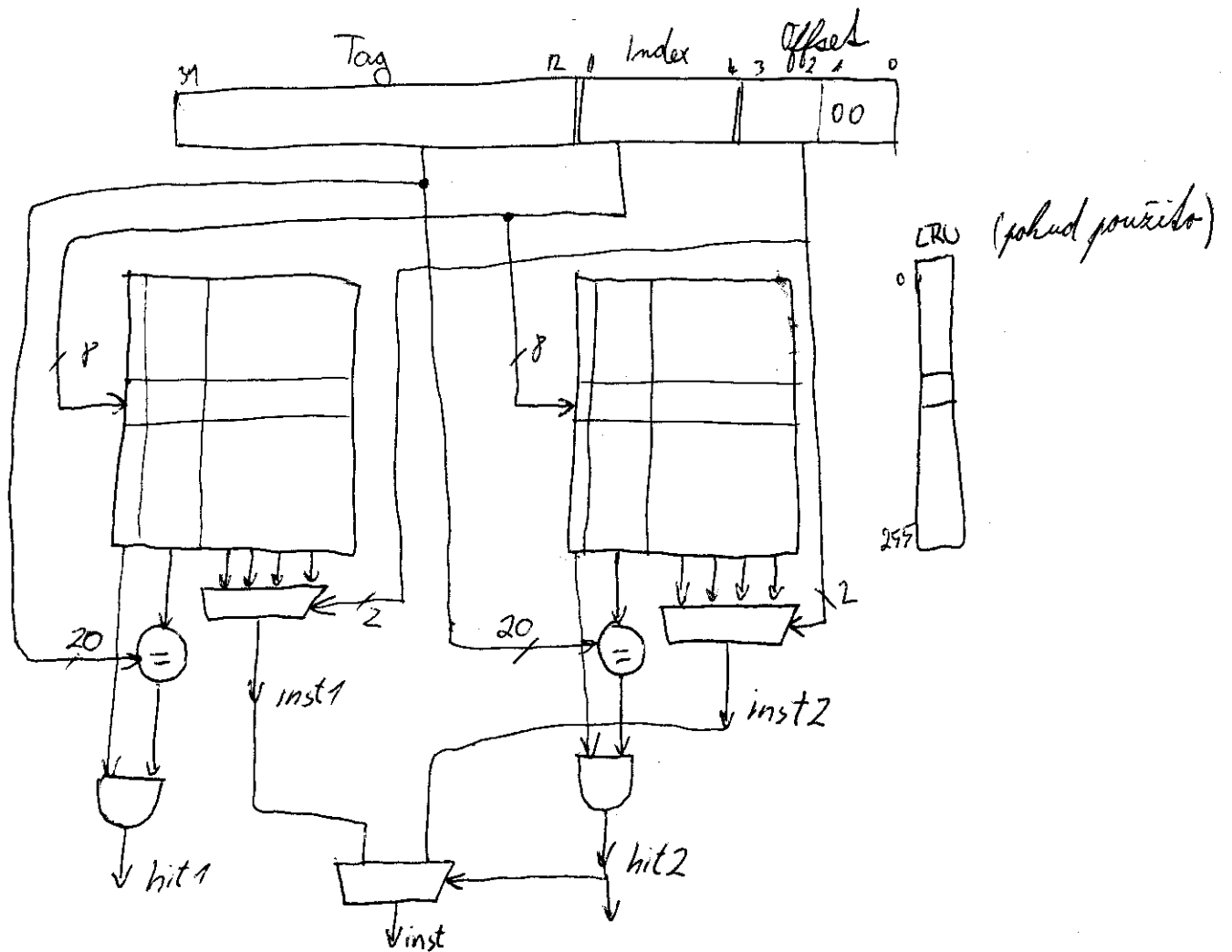
vel. slova 4 B (32 bit = 1 inst.), instrukce karovnané na 4 B ady.

vel. adresy 32 b.

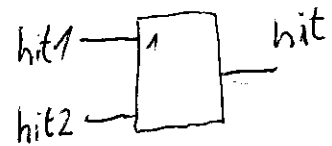


$$\frac{4 \text{ KB}}{16 \text{ B}} = \frac{2^{12}}{2^4} = 256$$

$$\text{Tag} = 32 - 8 - 4 = 20 \text{ b}$$



hit1	hit2	hit	inst
0	0	0	x
0	1	1	inst2
1	0	1	inst1
1	1	x	x

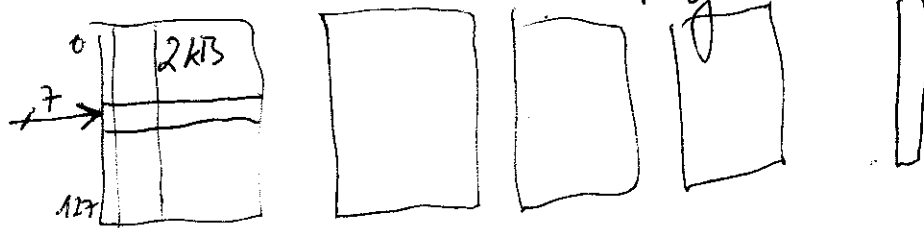


celková vel. $2 \cdot 2^8 \left(\underbrace{16}_{\text{data}} \cdot \underbrace{8}_{\text{tag}} + \underbrace{20}_{\text{index}} + \underbrace{1}_{\text{LRU}} \right) + 2^8$

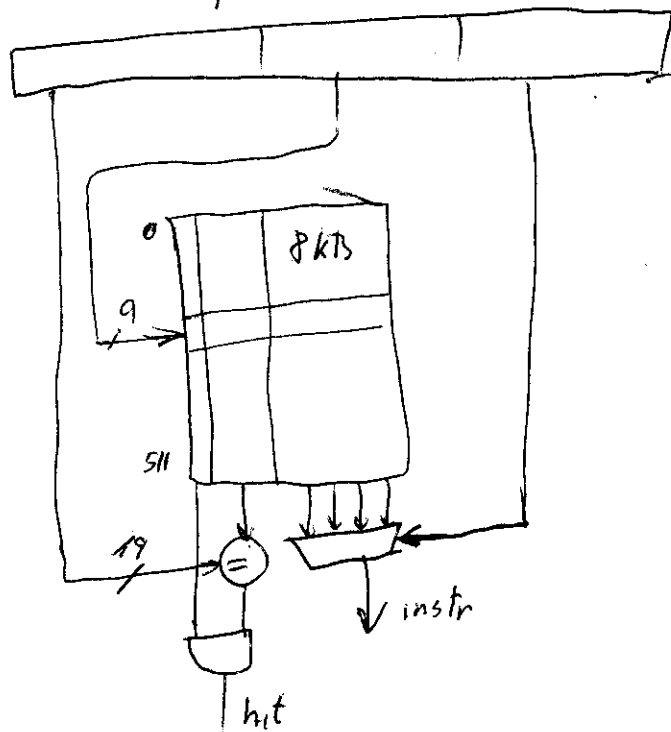
Změna

~~4096~~ ~~4096~~ $s=4$

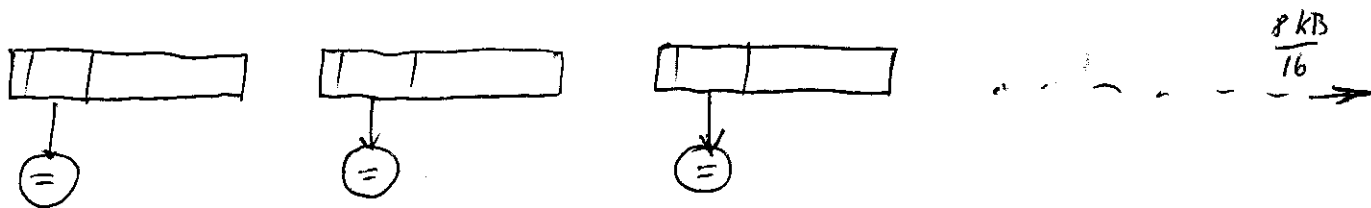
Index 7 bit
Tag + 1 bit LRU



Primo mapování



Plne asociativni
velky, slozity a pomaly -> pouze VELMI male cache



instanov L1 .. S=2
L2 .. S=4

PA

Program

PC	inst
0	a
4	b
8	c
20	f
24	g
4100	f
8200	0

Panel:

0	a	4096	i	8192	m
4	b	4100	j	8200	n
8	c	4104	k	8204	o
12	d	4108	l	8208	p
16	e				
20	f				
24	g				
28	h				

	inst	tag	index	offset	hit/miss
1)	a	0	0	0	miss
2)	b	0	0	1	hit
3)	c	0	0	2	hit
4)	f	0	1	1	miss
5)	g	0	1	2	hit
6)	f	1	0	1	miss
7)	0	2	0	2	miss

index off x
10 100

	v1	tag1	data1
0	0	x	x
1	0	x	x

	v2	tag2	data2	LRU
0	0	x	x	8
1	0	x	x	8

1), 2), 3)

0	1	0	a b c d
1	0	x	x

0	x	x	0
1	x	x	x

4)

0	1	0	a b c d
1			

0	0	x	x	0	0
1	1	0	~~a~~ e f g h	1	1

5)

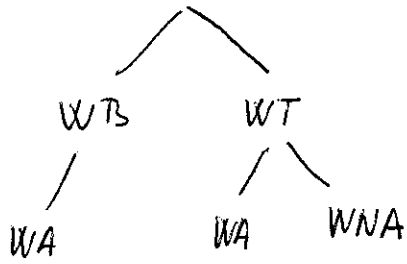
1	0	a b c d
0		

1	1	i j k l	1
1	0	e f g h	1

1	2	m n o p
---	---	---------

1	1	i j k l	0
1	0	e f g h	1

Datová cache

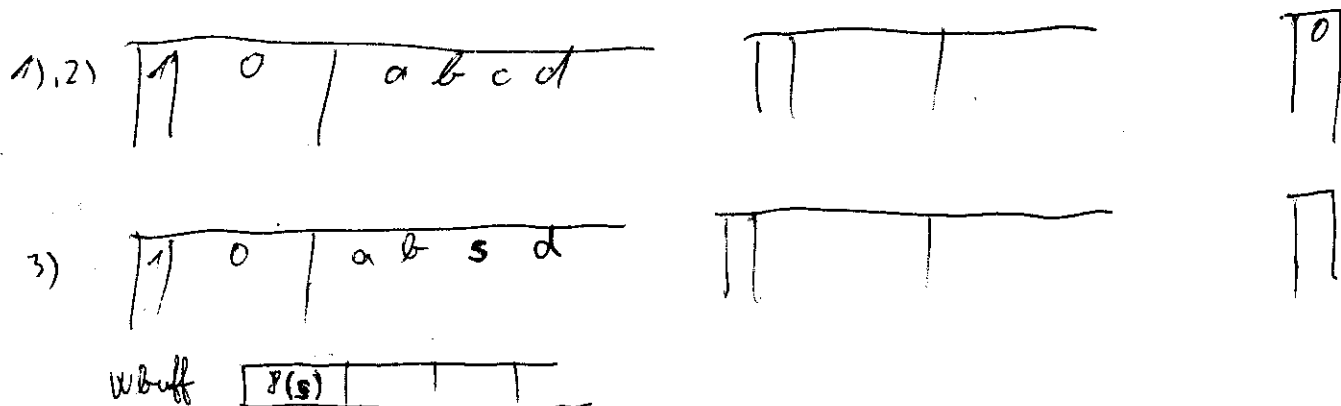


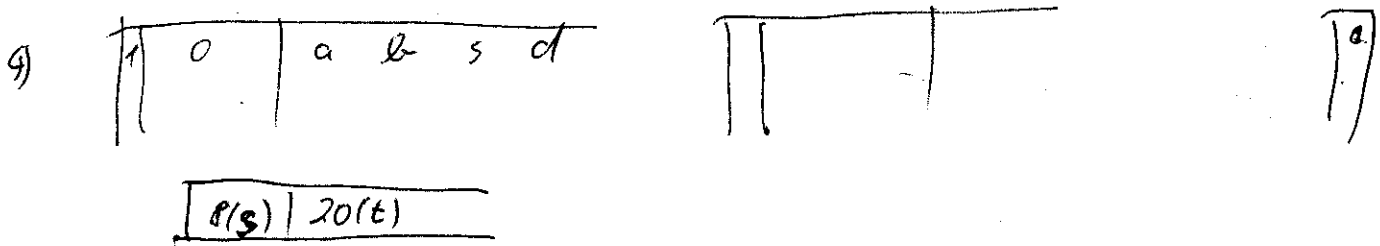
WA write allocate

Pr. — stejné jako minulý, jen WT, WNA

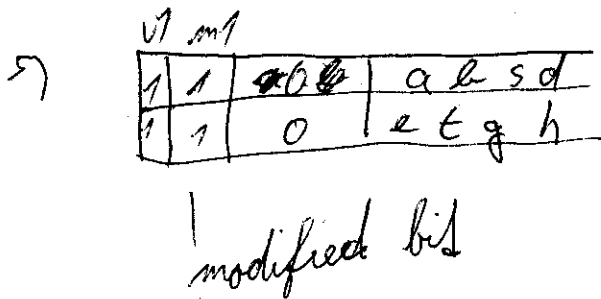
r c (w(s))
20 f (w(t))

	inst	tag	index	off	
1)	a	0	0	0	R miss
2)	b	0	0	1	R hit
3)	c	0	0	2	W hit
4)	f	0	1	1	W miss
5)	g	0	1	2	R miss





WB



5)	g	0	1	2	Rhit
6)	j	1	0	1	Rmiss
	o	2	0	2	Rmiss

$$AMAT = \frac{\text{Hit Time}}{HT + MR \cdot MP}$$

$$T_{CPU} = IC \cdot CPI \cdot T_{CLK}$$

$$\downarrow$$

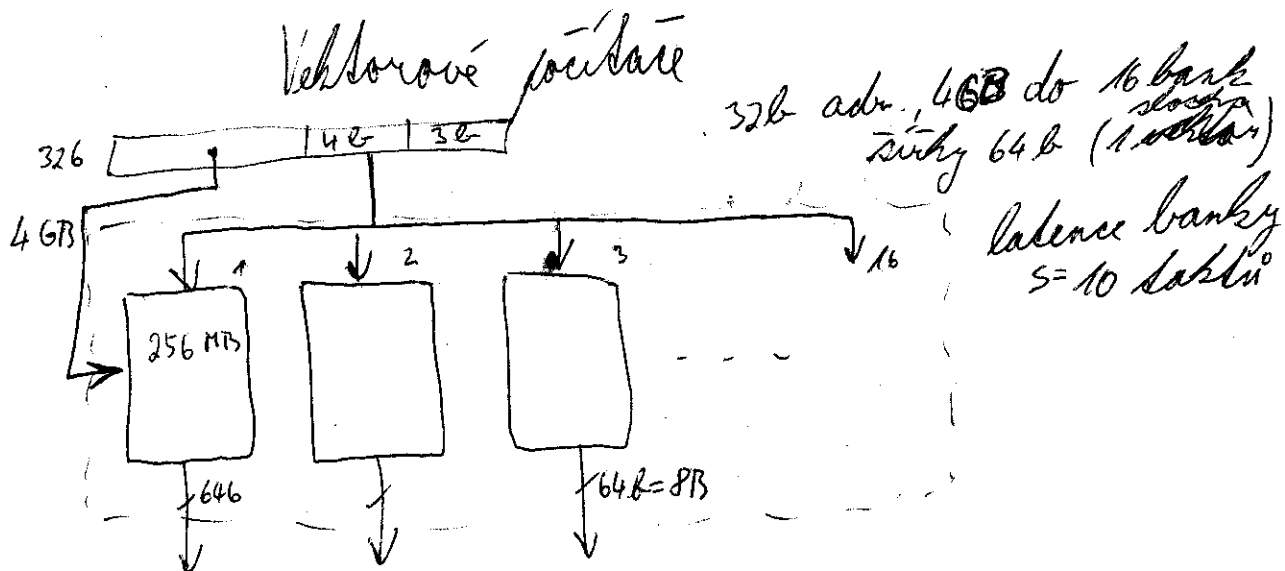
$$CPI_{IDEAL} + SP1 + MAP1 \cdot MR \cdot MP$$

$$(RISC) \quad MAP1 = 1 + LSR = 1,23$$

pare Trada

spice

Pr.



$R1 = 40A0$

— kolik takto bude brvat LV $R1, R1$

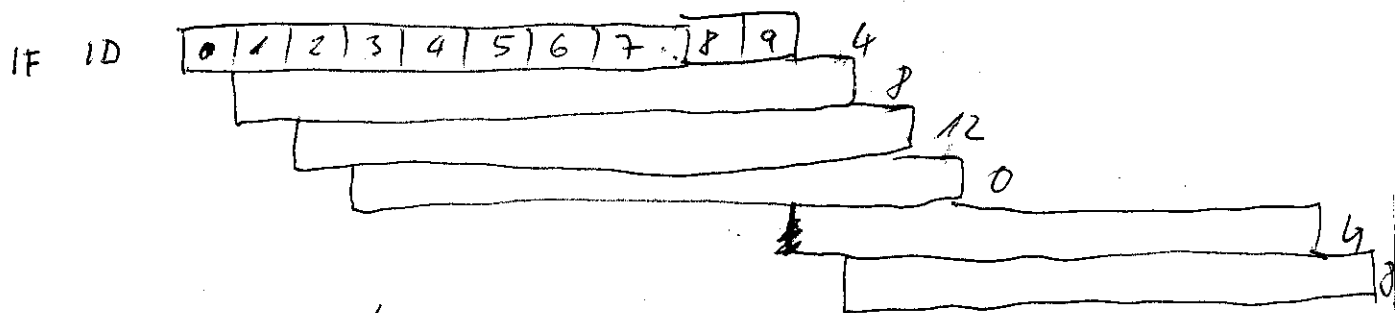
0100	0000	1010	0000
		0101	
		0110	
		...	

IF	ID	0	1	2	3	4	5	6	7	8	9	bank 4
												bank 5
												bank 6

~~bank 4~~

— a LVWS $R1, R1, 32$

0100	0000	1010	0000	4
		1000		8
		1100		12
		...		0
		...		4
		...		8



$$10 + 63 + \left(\frac{64}{4} - 1 \right) \cdot 6$$

$\downarrow$ $\downarrow$
 64x přístup, 6 sakků
 každý 4. prodloužení

Pr. 2

vektory X, Y 64 prvků, init interval LSU 14 sakků,
 sčítáčky 5 sakků, násobičky 6 sakků. Kolik sakků
 bude spotřebováno

```
for (i=0; i<64; i++) {
  Y[i] = C * X[i] + Y[i];
}
```

na vektorovém proc

a) - bez chainingu

b) - " - " -

c) - " - " - , bez zpracování 2 load a 1 store
 najednou

be zpracovával 2 load a 1 store

ADDUI R1, R0, X
 ADDUI R2, R0, Y
 ADDUI R3, R0, C

LV V4, (R1)

MULTSV V4, R3, V4

LV V5, (R2)

ADDV V4, V4, V5

SWV (R2), V4

konvoj

odhad

"chime"

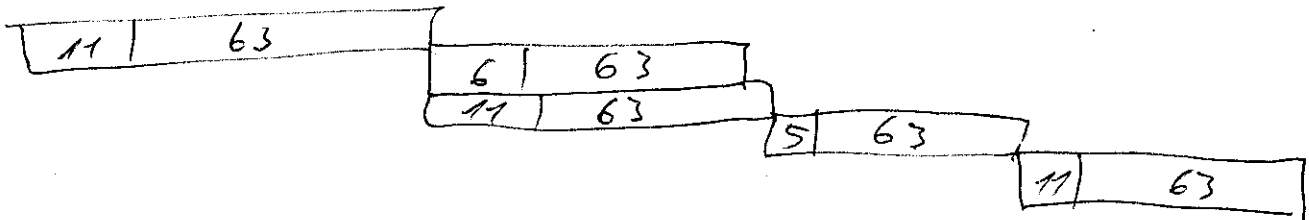
$$= 4 \cdot 64 = 256$$

1

2

3

4



$$74 + 74 + 68 + 74$$

2) chainingem

forwarding výsledků, odpadnou datové hazardy

LV

MULTSV

1

"chime"

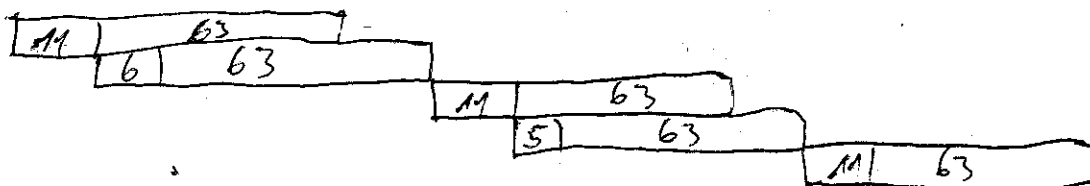
$$= 3 \cdot 64 = 192$$

LV

ADDV

2

SWV



$$11 + 6 + 63 + 11 + 5 + 63 + 11 + 63$$

c) 2x load

LV

MULT SVV

LV

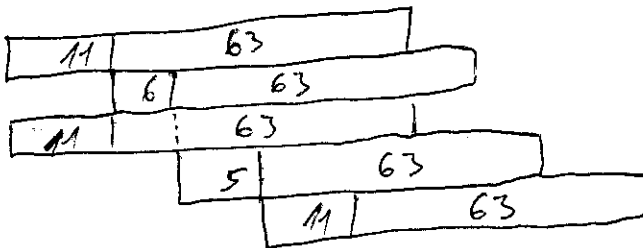
ADDV

SVV

1

"times"

$$= 1 \cdot 64 = 64$$

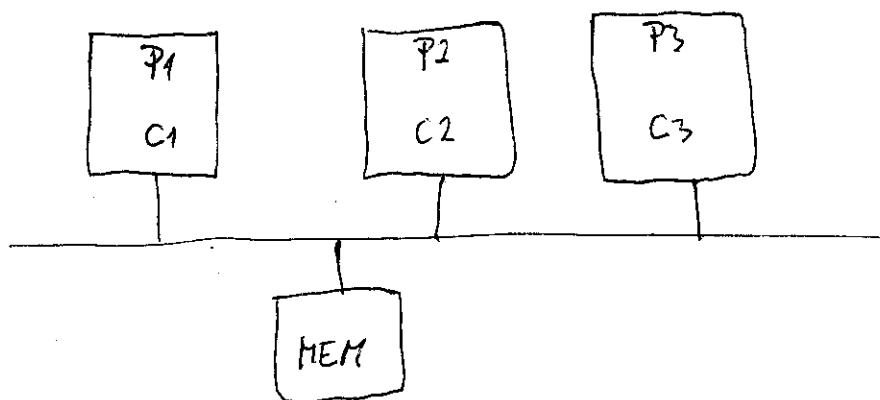


$$11 + 6 + 5 + 11 + 63 = 96$$

Pr 3

ULR = 8, VM range 1

SMP



koherence se odvíjí k jedné paměťové buňce,
konzistence k celé cache

P1 Rd $M(x)$
 Wr $M(x), e$
 Wr $M(x), g$

P2 Rd $M(x+1)$
 Wr $M(x+1), f$

P3 Wr $M(x+2), g$

HP

a	b	c	d
x	x+1	x+2	x+3

	P1 stav koda	P2 stav koda	P3 stav koda	ll. pam	op. na s. strán
Poč. stav	1	1	1	a b c d	
P1 Rd $M(x)$	V a b c d	1	1	a b c d	Bus Rd
P2 Rd $M(x+1)$	V a b c d	V a b c d	1	a b c d	Bus Rd
P3 Wr $M(x+2), g$	1	1	1	a b g d	Bus Wr
P1 Wr $M(x), e$	1	1	1	e b g d	Bus Wr
P2 Wr $M(x+1), f$	1	1	1	e f g d	Bus Wr
P1 Wr $M(x), g$	1	1	1	g f g d	Bus Wr

1... invalid, V... valid

	P1	P2	P3	hl. pam	op. na s. sb.
Proc. star	1	1	1	a b c d	
P1 Rd M(x)	V a b c d	1	1	a b c d	Bus Rd
P2 Rd M(x+1)	V a b c d	V a b c d	1	a b c d	Bus Rd
P1 Wx M(x), e	V e b c d	1	1	e b c d	Bus Wx
P2 Wx M(x+1), f	1	1	1	e f c d	Bus Wx
P1 Wx M(x), g	1	1	1	g f c d	Bus Wx
P3 Wx M(x+2), g	1	1	1	g f g d	Bus Wx

MESI	P1	P2	P3	hl. pam	op. na s. sb.
Proc. star	1	1	1	a b c d	
P1 Rd M(x)	E a b c d	1	1	a b c d	Bus Rd (Mem Sup.)
P2 Rd M(x+1)	S a b c d	S a b c d	1	a b c d	Bus Rd (Mem Sup.)
P1 Wx M(x), e	M e b c d	1	1	a b c d	Bus Upgr
P2 Rd M(x+1)	S e b c d	S e b c d	1	e b c d	Cache Flush
P3 Wx M(x+2), f	1	1	1	e b f d	Bus Rd X
P1 Wx M(x), g	1	1	1	e b f d	Bus Rd X / Cache Flush