APO FOR DUMMIES <3

příklady ze zkouškových testů



(já bych nerad, abyste tvořili databázi testů a učili se z toho)

OBSAH

PŘEVODY FLOAT IEEE754

STRÁNKOVÁNÍ

ALU SČÍTAČKA

DOBA VYKONÁNÍ PROGRAMU

CACHE ? (pls help)

ARGUMENTY FUNKCE

PROPUSTNOST V MIPS

FORWARDING

URČENÍ CPI

DATOVÝ PŘENOS NA SBĚRNICI PCI

HAZARDY ?(pls help)

SIGNÁLY PROCESORU (obrázky s odpověďmi)

TOTO JE ODHADEM NÁVOD K ASI 80 % TYPŮ PŘÍKLADŮ V TESTECH

doporučuji projít si i sesbírané teoretické otázky s odpověďmi, je to na disku

o chybách mě prosím informujte i v zájmu ostatních, kdo se na zkoušku učí :--))))

PŘEVODY FLOAT (IEEE 754)

Číslo je reprezentováno ve formátu:

znaménko	exponent	mantisa

kde znaménko = 1 bit, exponent = viz zadání úlohy, mantisa = viz zadání úlohy.

1) převod čísla

Příklad: exponent 4 bits, mantisa 5 bits.

Máme převést číslo 5,71.

postup:

1. Napřed vyjádříme celé číslo binárně. 5₁₀ = 101_b

2. vezmeme si desetinnou část čísla - 0.71. Desetinnou část budeme násobit dvěma a to tolikrát, kolik bitů má naše mantisa - tedy v tomto případě pětkrát. Z výsledku oddělíme celou část, to je náš výsledek hodnoty bitu a s desetinnou částí opakujeme proces.

$$0.71 \times 2 = 1.420.42 \times 2 = 0.820.84 \times 2 = 1.680.68 \times 2 = 1.360.36 \times 2 = 0.72$$

3. Nyní dáme obě části dohromady

101.10110

- 4. Desetinnou tečku posuneme o tolik pozic doleva, aby před desetinnou tečkou byla jen jedna jednička. V našem případě o 2 pozice.
- 5. Zjistím hodnotu exponentu

Exponent se rovná počtu pozic, o kolik jsme v minulém kroku posouvali + aditivní konstanta.

Aditivní konstanta: , kde a = počet bitů pro exponent (viz zadání).

V tomto příkladu $k = 2^{4-1} - 1 = 8 - 1 = 7$, exponent

= 2 + 7 = 9, binárně 1001.

- 6. Zbývá dopsat mantisu. Tu určím tak, že zapíšu čísla, která mám za desetinnou tečkou po posunu tedy opisuji čísla z kroku 4. Beru prvních 5 čísel, protože 5 je moje velikost mantisy (opět tato velikost je v zadání).
- 7. Znaménkový bit 0, pokud je číslo kladné, 1, pokud je číslo záporné.
- 8. Výsledek:

5.71 = 0100101101

0 - znaménko

1001 - exponent

01101 - mantisa

2) Jak bude v této reprezentaci zakódováno největší možné zobrazitelné reálné číslo?

Znaménko bude kladné. Poslední bit vpravo exponentu nastavíme na nulu, jinak všechny ostatní bity budou nastavené na 1 (kromě toho znaménka samozřejmě, nechceme záporné číslo).

Pro náš příklad - 4 bity exp, 5 bitů mantisa to bude vypadat takto:

0111011111

3) Jaká bude dekadická hodnota tohoto největšího čísla?

Převedeme z binárky zpět:

1. Určíme hodnotu exponentu

 $1110_2 = 14_{10}$, od tohoto čísla odečteme aditivní konstantu, 14 - 7 = 7. Čili náš exponent bude 2^7 .

2. Určíme hodnotu mantisy

Sčítáme postupně

2^7 * 63/32

$$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} = 31/32$$
 pro mantisu)

(v našem případě až do 2^{-5} , protože máme 5 bitů

3. Vynásobíme exponent s mantisou, ke které jsme přičetli jedničku



(iné riešenie: po zistení exponentu 2⁷ si vezmeme mantisu a dopíšeme k nej "skrytú jednotku" -> 1.11111. V binarnej sústave je násobenie dvoma bitový posun, teda 2⁷ je posun o 7 miest do prava. Dostaneme 1111 1100 = 252)

4) Jak bude zakódováno nejmenší nenulové kladné normalizované číslo?

Je to přesně opačně než při rešení otázky 2).

Tedy poslední bit exponentu nastavíme na 1, všechny ostatní budou 0.

Pro náš příklad, kde exponent 4 bits, mantisa 5 bits

0000100000

5) Jaká bude hodnota tohoto nejmenšího čísla?

Převedeme bity pro exponent:

0001_b = 1₁₀, odečteme aditivní

konstantu 1-7 = -6.

Exponent = 2^{-6}

Hodnota nejmenšího čísla je 2⁻⁶ = 0.015625, zaokrouhlíme 0.016

6) Pokud uvažujeme dekadické číslo x a aditivní konstantu y z-bitová reprezentace takto kódovaného číslo v kódus posunutou nulou?

Sečteme x a y a to poté vyjádříme binárně v dané bitové reprezentaci

STRÁNKOVÁNÍ

Ze zadání úlohy potřebuji vyčíst:

- -počet úrovní
- -velikost stránky
- -kolik bytes zabírá jedna položka

notlivých byte, orga položka 2-úrovňové části tabulky, tak v	Laburk	y pro	preki	ad vir	tuáln	adre	sy zab	oírá pr	ávě 21	yte.	Přiton	n před	poklá	deite	že in	k walilea	at a dans (×
části tabulky, tak v toru/paměti:	lastní	tabul	ky od	kazů i	na str	ánky	odpov	vídá p	rávě j	edné s	stránc	e. Vv	pis ol	osahu	fyziel	cho ad	reenthe an
toru/pameti:													P.0 0.	Journa	1 J BICE	cuo au	resmino pr
	x0:	x1:	x2:	x3:	x4:	x5:	x6:	v7.	v8.	200	A -	D	-	-	_	xF:	
00:	87	99	C5	9C	23	E0	I TO	I TOTO	AO.		XA:	XB:	XC:	xD:	xE:	xF:	
10. R	*CA		17	1		~	F8	EB	A5	BA	2F	FC	EB	2E	A9	3E	
			-	39	5D	C3	D3	94	FD	48	59	7C	E3	CA	-		- marine
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC				
30:	61	8A	72	30	A7	5A	A7	41	2C				-	E9	00	CC	
40:	(CD	4E	*80	00	4E	0.00				51	81	39	2C	DE	BC	18	
					-	EE	9A	41	62	D5	1B	92	CE	65	F8	6A	
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C					
60:	81	31	10	CE	01	730	EA			77.7	20		E1	5C	49	30	

E9 BD 9D CB DA AF 42 8A E8 64 80: 23 18 2C 10 •10 90: 3A 6C 4C DC CD 8C C1 E7 46 1E 2C 52 A0: AB 4B A3 AF C2 EC 7E 59 3A 85 D2 B2 10 5C F5 4E 7D C8 39 2E 57

Nejvyšší úroveň stránkovací tabulky (hodnota PDBR – page directory base register) začíná na adrese 0x0040. Hodnota položky stránkovací tabulky vyjma nejnižšího bitu přímo určuje adresu. Nejnižší bit položky stránkovací tabulky určuje, jestli položka platná (0-platná, 1-neplatná). Procesor čte z virtuální adresy 0x00DE.

1) Kolik bitů virtuální adresy je jednotka správy s výše uvedenými parametry schopna přeložit?

1. velikost stránky vyjádřím jako 2ⁿ

Př. vel. stránky =
$$16 B = 2^4$$

číslo n určuje, kolika bity je určen offset, tady je určen čtyřmi bity

2. zjistím, kolik položek se vejde do stránky, opět vyjádřím jako 2ⁿ

$$16/2 = 8 = 2^3$$

index stránky je tedy určen třemi bity

3. kolik mám úrovní, tolikrát vynásobím bity pro index

4. výsledek

$$4 + 3 + 3 = 10$$

S uvedenými parametry je schopna přeložit 10 bitů.

2) Jaký je offset v rámci fyzické stránky?

Virtuální adresu ze zadání si přepíšu do binární soustavy 0x00DE = 0000 0000 1101 1110 offset mi určují 4 bity, vezmu poslední 4 bity od konce - 1110, to je v dec 14

3) Jaký je index v tabulce stránek první (a druhé) úrovně?

Virtuální adresu ze zadání si přepíšu do binární soustavy

0x00DE = 0000 0000 1101 1110 index

je určen třemi bity

Vezmu proto tři bity z binárního zápisu adresy, znázorním v obrázku pro jednoduchost:

0000 0000 1101 1110 červená -

offset (předchozí otázka) modrá -

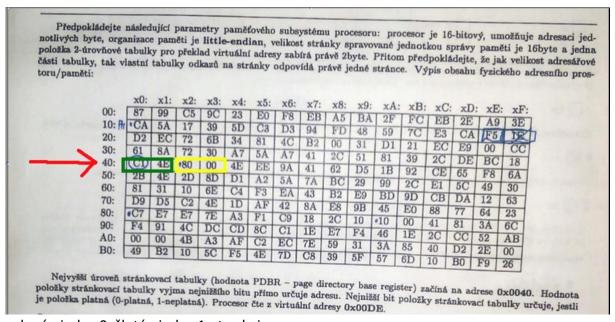
index 2. úrovně zelená - index 1.

úrovně převedu opět do dec:

index 2. úrovně = 5, index 1. úrovně = 1

4) Jaká data budou z adresy načtena?

- 1. vezmu startovací adresu (v zadání) 0x0040
- 2. posouvám se vpravo o index 1. úrovně. Pozor, posouvám se po dvojících, jelikož velikost položky je zrovna 2 B. Začínám na indexu 0.



zelená - index 0, žlutá - index 1 - to chci

3. organizace paměti je little endian, musím tedy prohodit pořadí obdélníčku ve dvojici 4. dostávám adresu 0x0080, na tento řádek se posunu a opakuji postup - tedy posouvám se směrem vpravo o index 2. úrovně

Předpokládejte následující parametry paměťového subsystému procesoru: procesor je 16-bitový, umožňuje adresaci jednotlivých byte, organizace paměti je little-endian, velikost stránky spravované jednotkou správy paměti je 16byte a jedna položka 2-úrovňové tabulky pro překlad virtuální adresy zabírá právě 2byte. Přitom předpokládejte, že jak velikost adresářové části tabulky, tak vlastní tabulky odkazů na stránky odpovídá právě jedné stránce. Výpis obsahu fyzického adresního prosx1: x2: x3: x4: x5: x6: x7: x8: x9: xA: xB: xC: xD: xE: xF: 23 E0 F8 EB A5 BA 2F EB 2E 10: # CA C3 D3 94 FD 48 59 20: D2 EC 4C B₂ D1 21 EC E9 30: 30 A7 5A A7 81 2C DE BC EE 9A 41 62 92 65 F8 50: 6A 4E D1 5A BC 7A 29 99 2C 31 10 6E C4 F3 EA B₂ E9 BD 9D CB DA 70: 4E 1D AF 42 8A 9B 45 80: 64 23 E7 | 7E A3 C9 | 18 2C 10 3A 6C CDI 8C C1 2C52 AO: A3 AB AF C2 EC 7E 31 3A 40 D2 B0: 2E F5 4E 7D C8 39 5F 57 6D 10 F9 Nejvyšší úroveň stránkovací tabulky (hodnota PDBR – page directory base register) začíná na adrese 0x0040. Hodnota položky stránkovací tabulky vyjma nejnižšího bitu přímo určuje adresu. Nejnižší bit položky stránkovací tabulky určuje, jestli

opět prohodím kvůli little-endian, dostávám adresu 0x0010 5. nyní zbývá posunout se o offset.

Offset vždy posouvám pouze po jednom obdélníčku, startuje to opět na nule. Můj offset je 14 - tedy dostávám se na adresu 0x001E:

Předpokládejte notlivých byte, org položka 2-úrovňové	tabull	v pro	překl	ad viv	+udla	(nolon		or sma	шку в	prave	vane	jednot	kou s	právy	pam	ěti je 1	l6byte a
položka 2-úrovňové části tabulky, tak v toru/paměti:	lastní	tabul	ky od	kazů	na str	ánkv	odnov	ora pr	ave 2	byte.	Přito	m před	dpokla	idejte	, že ja	k velik	ost adres
toru/paměti:					-		capo	rua p	lave J	eane	strane	ce. V	ypis o	bsahu	fyzic	kého a	dresního
00	x0:	x1:	x2:	x3:	x4:	x5:	x6:	x7:	x8:	x9:	xA	: xB	xC.	vD.	. vF	xF:	
00:	87	99	C5	9C	23	E0	F8	EB	A5	BA	2F	FC			_	3E	1
	*CA	5A	17	39	5D	C3	D3	94	FD	48	59	7C	E3	CA		1 DE	-
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC	E9			_
30:	61	8A	72	30	A7	5A	A7	41	2C	51	81	39	2C	DE		CC	
40:	(C1)	4E	*80	00	4E	EE	9A	41	62	D5	1B	92	CE	65 65			
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C	E1		F8	6A	
60:	81	31	10	6E	C4	F3	EA	43	B2	E9	BD	9D	CB	5C	49	30	
70:	D9	D5	C2	4E	1D	AF	42	8A	E8	9B	45	EO		DA	12	63	
80:	•C7	E7	E7	7E	A3	F1	C9	18	2C	10	•10	00	88	77	64	23	
90:	F4	91	4C	DC	CD	8C	C1	1E	E7	F4	46	1E	41	81	3A	6C	
A0:	00	00	4B	A3	AF	C2	EC	7E	59	31	3A	85	2C	CC	52	AB	
B0:	49	B2	10	5C	F5	4E	7D	C8	39	5F	57	6D	40	D2	2E	00	
													10		F9	26). Hodnot

Velikost jedné položky je 2byte, tedy načtená data nejsou pouze F5 ale 0xF51E, jenže organizace paměti je little-endian tzn. Výsledek je, že se načtou data 0x1EF5

5) Na které fyzické adrese jsou uložena data z virtuální adresy 0xabcd?

Data jsme nalezli v předchozí otázce. Pokud tam taková otázka v zadání nebyla, provedeme proces viz otázka 4). Pro zjištění fyzické adresy se stačí podívat na souřadnice toho nalezeného obdélníčku.

Vidíme, že se nachází na adrese 0x001E (řádek 10, sloupec E) Pokud to otázka vyžaduje, převedeme adresu do binární soustavy.

6) Kolik úrovní by bylo potřeba pro stránkovací tabulku, aby byl pokrytý překlad z 16-bit virtuální adresy na 16-bit fyzickou adresu? (možná druhá varianta zadání - z 8-bit na 8-bit)

- 1. Z předchozích otázek již vím, kolik bitů potřebuji pro index stránky a kolik pro offset a kolik celkem je schopná jednotka paměti přeložit bitů.
- 2. Tato moje v zadání překládá 10 bitů a má dvě úrovně.
- 3. Přidáním další úrovně budu schopná přeložit 13 bitů. Ještě další 16 bitů. A to hledám. Musím proto přidat 2 další úrovně.
- 4. Celkem tedy potřebují mít 4 úrovně, aby byl pokrytý překlad z 16-bit virt. na 16-bit fyz.

(Poslední úroveň nemusí vyjít přesně na 16 bitů, může to být např. taková situace: Offset je 5 bitů, index 4 bity, mám dvě úrovně. Celkem tedy 5+4+4=13, chci dopočítat do 16 bitů. Přidám třetí úroveň - bude to 13+4, to už je víc než 16, takže mi v tomto případě budou stačit 3 úrovně. Ten součet musí být vždy větší roven 16 bitům.)

Druhá možná varianta zadání: z 8-bit na 8-bit - úvaha a postup stejný, jen nedopočítávám do 16, ale do 8 bitů

ALU SČÍTAČKA

Uvažujte 8-bitovou ALU dle obrázku vpravo. Vstupem jsou dvě 8-bitová binární čísla A=0b101111, B=0b11100100. Výstupem je 8-bitový součet A+B a indikace Z (zero), C (carry) a OV (overflow). (Pokyny pro vyplnění: Box zakřížkujte pouze pokud má být daný bit nastaven na 1. Prázdný box je považován za 0. Nejméně významný bit je úplně vpravo.)	ALU Z C C OV
Question 11 (+2b) Jaký bude výsledek součtu? 1 1 1 1 1 1 1	Question 12 $(+2b)$ Jak budou nastaveny příznaky Z, C a OV a bit F, pokud F realizuje logickou funkci $F = \overline{Z} \cdot \overline{C} \cdot \overline{OV}$? Boxy vyplňujte v uvedeném pořadí zleva doprava, tj. první box zleva představuje Z, druhý C, třetí OV a čtvrtý F.

Máme dvě 8-bitová čísla - A a B

Čísla sečteme: 101111 + 11100100 100010011

Jaký bude výsledek součtu?

-napíšeme tam, co nám vyjde (8 bitů zprava)

Jak budou nastaveny příznaky?

Z ... zero -Nastavený, pokud výsledek je nula

C ... carry -Nastavený, pokud mám vypůjčenou jedničku z místa, kam nedosáhnu (tedy jakoby devátý bit zprava)

OV ... overflow -Nastavený, pokud nejvýznamnější bit (znaménkový) změní svou

hodnotu - tj. pokud při sčítání dvou kladných čísel (znaménko 0) změní tento bit hodnotu na 1 F ... nastavený, pokud jsou všechny předchozí příznaky na nule, tedy negace logického součinu Z, C a OVs

http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt

DOBA VYKONÁNÍ PROGRAMU

Mějme procesor s nějakou frekvencí (určuje počet cyklů za 1 s)

Dále máme počet instrukcí a jejich dobu trvání a poměr, kolik z nich jsou přístupy do paměti a jejich trvání. Př.

akt vyjma instrukc lané paměťové insti	a printupu do panter. V	mci 1GHs a program čitají o jednoduchost předpoklád uvedeném programu 25% i a tento procesor uvedený p	(1000 instrukci. Vykonání lejte, že jeden přístup do p nstrukci jsou přístupy do p rogram?	každė instrukce trvė jeden amēti (epolu s vykonanim amēti. 15° cc c. 1 - po.m.či a 150° cc - o backo
20800ns 21500ns 51500ns	30500ns 32350ns 25750ns	41600ns 65450ns 37000ns	42500as 38000as	50000ns 45000ns 22450ns nā miss rate 50% pro daný 10ns.
program a doba na 5500ns 2600ns 4500ns	9300as 3200as 8000as	6000ns 5000ns 6500ns	3450ns 4000ns 6750ns	3000ns 3500ns 15750ne

f = 1 GHz n = 1000 p = 25 % t1 = 1 takt - 1 ns t2 = 100 ns

1) Za jakou dobu vykoná tento procesor uvedený program?

$$250 \times 100 + 750 \times 1 = 25750$$
ns

2) Jak se změní tato doba, když přidáme do systému cache paměť?

Ze zadání vzčteme miss rate a dobu nalezení záznamu cache.

Nyní každá instrukce, která má přístup do paměti, bude hledat v cache - tedy bude trvat dobu nalezení záznamu.

Miss rate je 50 %, tedy polovina z těch přístupů do paměti se v cache nenajde, takže budou trvat dalších 100 ns, jako v předchozí otázce.

$$250 \times 10 + 750 \times 1 + 125 \times 100 = 15750$$
ns

CACHE

Uvažujte datovou cache s následujícími parametry – velikost cache: 2048B, dělka bloku: 64B, stupeň asosiativity: 1. Cache je na začátku prázdná. Dělka slova je 32 bitů. Předpokládejte, že program čte sekvenčně v jednom cyklu z paměti počínaje od adresy 0x33C právě 692 slov. Počet opakování cyklu je 7, takže procesor čte celkem 692×7 slov. Souběh jiných programů nepředpokládáme.

Potřebuji parametry: stupeň asociativity velikost cache délka bloku délka jednoho slova

1) Kolik má cache setů?

Spočítám podle vzorce

počet setů S =
$$\frac{\text{velikost cache}}{\text{délka bloku} \times \text{stupeň asociativity}} = \frac{2048}{64 \text{ x 1}} = 32 \text{ setov}$$

2) Uveďte číslo setu, do kterého se namapují data z adresy 0x33C.

Adresu převedu na decimální číslo = 3 x 256 + 3 x 16 + 12 = 828 vydělím adresu počtem slov, která se vejdou do jednoho bloku x 4

-1 slovo má 32 bitů, délka bloku 64 B -> 64/4 = 16

828 / (4 x 16) = 12,9375

-> z toho beru jen to celé číslo

a teď modulo počet setů:

Takže data se namapují do setu číslo 12.

Iný spôsob:

Dĺžka bloku je 64B, teda offset bude 26 (6 bitov).

Počet setov: 2048/64 = 32, teda na zakódovanie pozicie v setoch bude 25 (5bitov).

Keďže začiname na adrese 0x33C vieme set zistiť ako:

Teda vidíme, že sme v 12tom sete a čítame od pozície 60.

3) Počet missů/hitů cache při prvním průchodu 1cestná

Do cache se bude načítat postupně 692 slov (velikost 692 slov je vyšší než velikost cache, to ale v prvním průchodu nehraje roli)

Slova se budou do cache načítat po blocích, resp. Po setech, tzn načteme jedno slovo, ale do cache se ve skutečnosti načte počet slov v jednom bloku/setu – 64*8/32 = 16 Tedy k missu dojde pouze pokud se ptáme na první slovo v tomto setu. Aj keď čítame slovo zo stredu bloku načítajú sa aj slová v bloku pred ním. Z predchádzajúcej ulohy podľa offsetu vieme vyčítať čítanie z pozície 60. Teda prve slovo nám načíta set 12 no my z neho len

posledné slovo a dostaneme prvý miss. Následne ostáva prečítať už len 691 slov \rightarrow 691 / 16 = 43,1875 \rightarrow 44x budeme načítat do cache

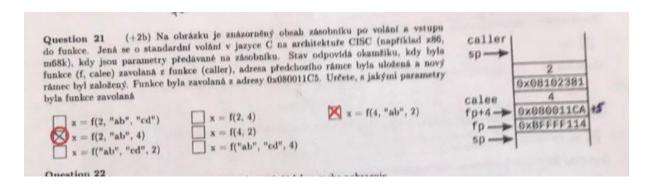
Tedy 1 + 44 missů a 692 – 45 = 647 hitů

4) Celkový počet missů/hitů cache 1 cestná

Je potřeba si uvědomit jak načítání probíhá, máme cache o velikosti 2048*8/32 = 512 slov, jenže načítáme 692 slov. Tedy rozdíl je 180 slov → 12 setů, pokud by velikost cache byla větší rovna celkové velikosti načítaných slov, celkový počet missů zůstane 45. Tedy při každém dalším cyklu si budeme v cache přepisovat 12 setů, tzn. Přepíšeme je jednak na začátku cyklu, jednak i na konci cyklu, 12 * 2 = 24 setů budeme v každém dalším cyklu znovu načítat do cache,

Výsledek \rightarrow 45 missů z prvního cyklu + 24 missů * 6 počet dalších cyklů = 189 missů celkem celkový počet hitů \rightarrow 692 * 7 – 189 = 4655 hitů celkem

ARGUMENTY FUNKCE



Všimněme si v obrázku, kam směřují popisky.

Argumenty hledáme mezi řádky "calee" a "sp".

Jsou tam tři - budou tři argumenty funkce.

Pořadí je od spodu směrem hore, takže 4, "něco", "2".

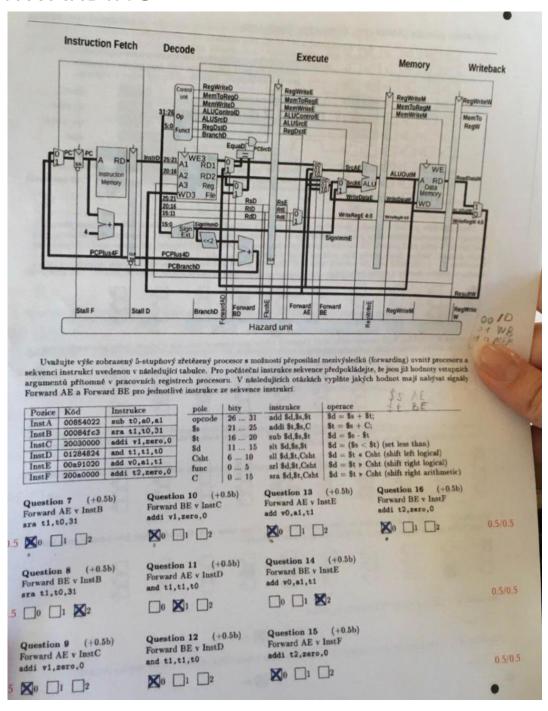
To něco zpravidla je nějaký string, ale podle čísel vybereme možnost v zadání.

PROPUSTNOST V MIPS

ID: 6ns, EX: 3	e průměrná propustne ních závislostí je v pr Víme, že latence jedne 35ns, MEM: 6ns, WB:	otlivých stupňů (komb 3ns a frekvence proce	diziter hyen the deposited	upňové logiky) proce naximální možnou. Z	second), pokud v důsledku a (bubble – jako důsledek soru je následující IF: 9ns, poždění v důsledku čekání před hodinovým signálem dbejte.
☐ 39 ☐ 29 ※ 24	☐ 27	☐ 47	28	□ 0 □ 28	6
24	☐ 26 ☐ 5	☐ 10 ☐ 0	23 13	34	1
				10.11	

Zjistím poměr užitečných instrukcí / bublina ze zadání: každá sedmá "instrukce" je bublina tedy poměr je 6/7 Spočítám frekvenci procesoru: najdu, která fáze má nejvyšší latenci, vezmu hodnotu této latence a spočítám, kolikrát se vleze do 1 s vyjde mi frekvence v Hz, tu převedu do MHz a vynásobím poměrem
1 s / 35 ns = 28,6 MHz
28,57 MHz x 6/7 = 24.48 = 24

FORWARDING



Máme dva signály - AE a BE. signál AE
je pro registr \$s, signál BE je pro registr
\$t (viz instrukce vpravo vedle kódu)
Závislost je hned pod sebou - signál 2,
závislost je ob jeden řádek - signál 1,
mezera je víc než jeden řádek - signál 0.
Pokud má instrukce konstantu a daný registr (\$s nebo \$t) nepoužívá, dáme do signálu 0.
V obrázku jsou všechny hodnoty správné.

URČENÍ HODNOTY CPI

misss. Určete ce	s per instruction) rovi	no 4 bez jakýchkoliv	memory stalls. Dále	uvažujte penalizaci 10	cache 4%. Procesor ma 00 cyklú pro každý cache představují load a store
instrukce.					
instrukce.	7.40	□ 6.82	0.40	15.99 11.00	5.19 8.85

Ze zadání zjistíme:

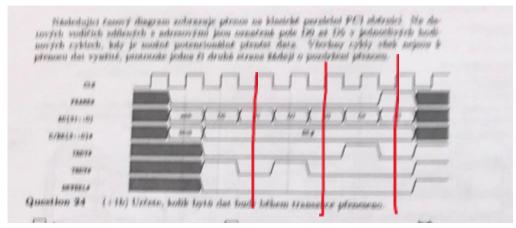
default CPI = 4 miss rate instr. cache = 6 % miss rate dat. cache = 4 % za cache miss penalizace = 100 cyklů 25 % instrukcí jsou load a store

25 % jsou load a store -> 25 % jsou přístupy do paměti, ve 25 % tedy využíváme datovou

cache instrukční cache využíváme vždycky, takže 100 %

CPI = defCPI + (miss inst. x 1 x penalizace) + (miss data x 0.25 x penalizace) CPI = $4 + (0.06 \times 1 \times 100) + (0.04 \times 0.25 \times 100) = 4 + 6 + 1 = 11$

DATOVÝ PŘENOS NA SBĚRNICI PCI



Máme diagram signálů.

K datovému přenosu dochází, když jsou signálu IRDY#, TRDY# a DEVSEL# (poslední tři) na spodní hodnotě (na nule!) a zároveň CLOCK (první) je na hraně Vyznačeno v obrázku.

1) Určete, kolik bytů dat bude během transakce přeneseno.

-vezmeme počet přenosů a vynásobíme čtyřmi, protože na třetím signálu odshora je napsané AD(31: sth dunno) -> 31 značí, že to budou 4 byty - 32 bitů sorry for bad image (")/ $^-$

3 x 4 = 12 -bude přeneseno 12 Bytů

2) Hodnoty vodičů označené D0 (D1, D2, ... Dn) představují okamžik, kdy jsou přenášená data

ANO/NE

Na třetím signálu odshora jsou napsané ty D0, D1, D2 atd.

Pokud přenos (vyznačen červeně v obrázku) prochází v obrázku tou hodnotou D0, D1, D2, ... na třetím signálu, tak volím ANO, jinak NE

HAZARDY

	mačeno). Do tabi	1	2	3	4	5	6	17	8	9	10	11	12	13	14	15	16	17
	sub s4,s1,t1	IF	ID	EX	M	WB												-
	lw t2,0x4(s4)		-														-	-
	addi s2,t1,0x3					-							-		-	-		
	add s4,s4,0x4						-					-	-	-		-		
	sw s2,0(s4)					ude př						lat inc	tenko	o z tal	mlky?	,	-	
Qu	7	. 🛛	5			oude p	16 8				11 15			9 13				12
_																		

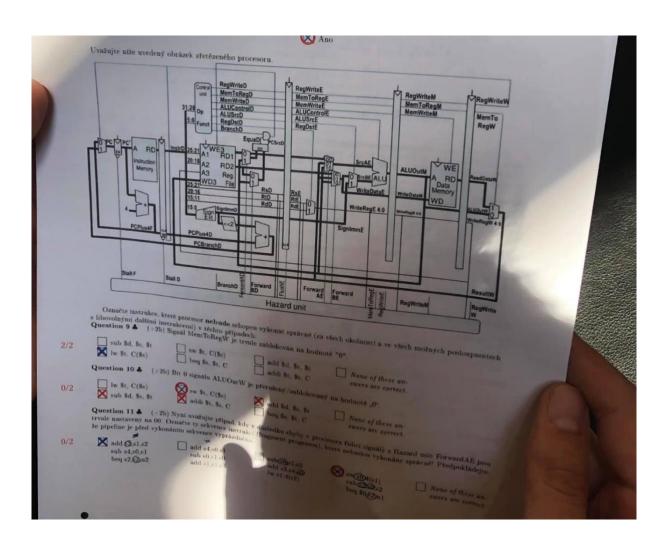
dvě možnosti zadání - procesor s možností forwardingu, nebo procesor bez možnosti forwardingu

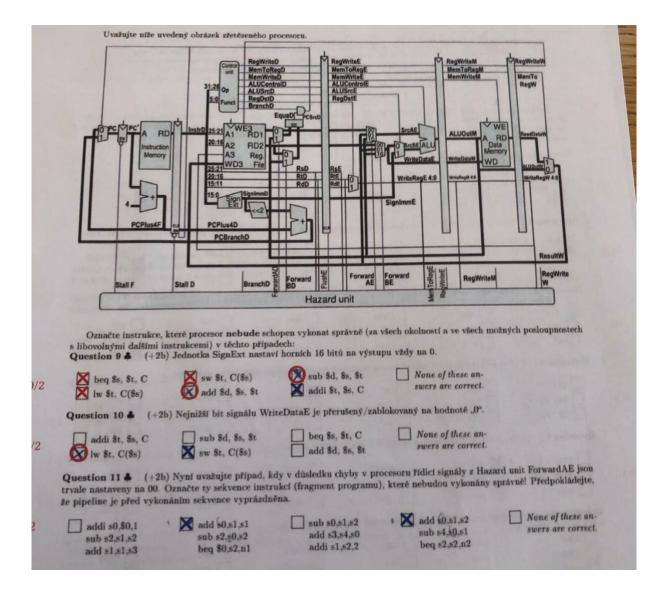
bez forwardingu - easy, přidávám jeden nebo dva stally tam, kde je závislost mezi registry **s forwardingem** - záleží na typu instrukcí

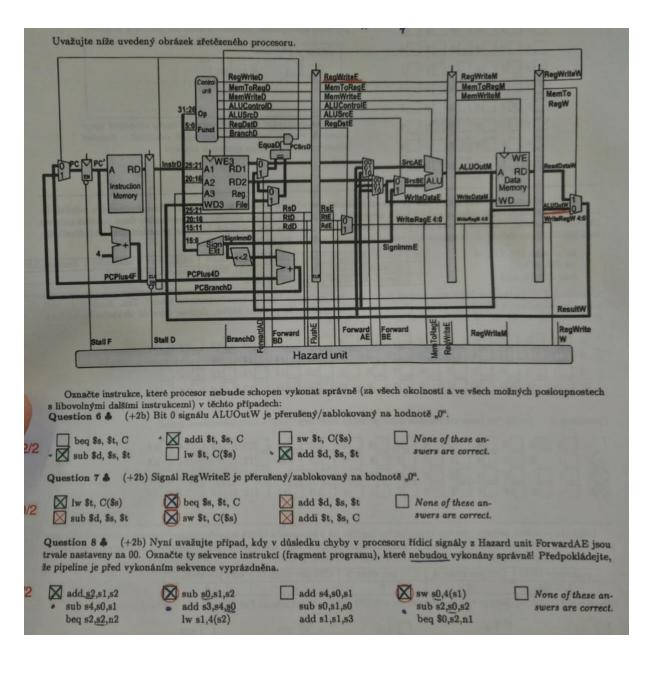
- -mezi <u>aritmetickými instrukcemi</u> forwarding vyřeší všechno, nemusím dávat žádné stally
- -mezi aritmetickou a paměťovou (lw, sw) musím dát jeden stall, pokud je tam závislost
- -<u>branch instrukce</u>: pokud instrukce skoku zavisi na instrukce před ní, musí nastat stall, protože branch instrukce se vyhodnocuje ve fázi ID a není možné přeposílat z fáze EX do ID

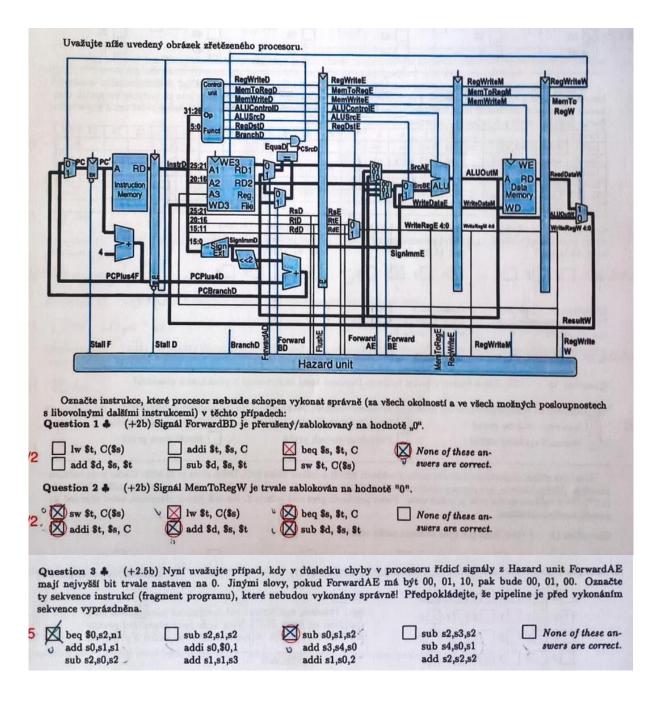
btw. není to sice opravené, ale odpovědi jsou správné, můžete si to ověřit pomocí QtMipsu

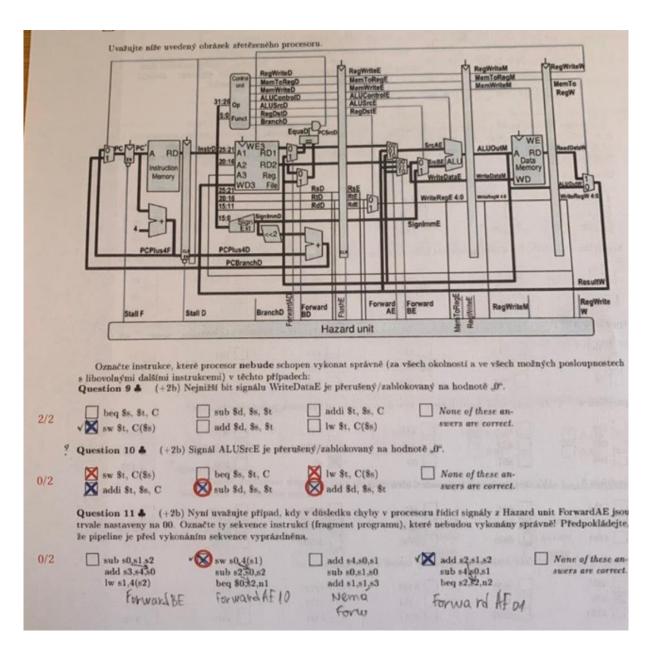
BETTER LEARN THIS

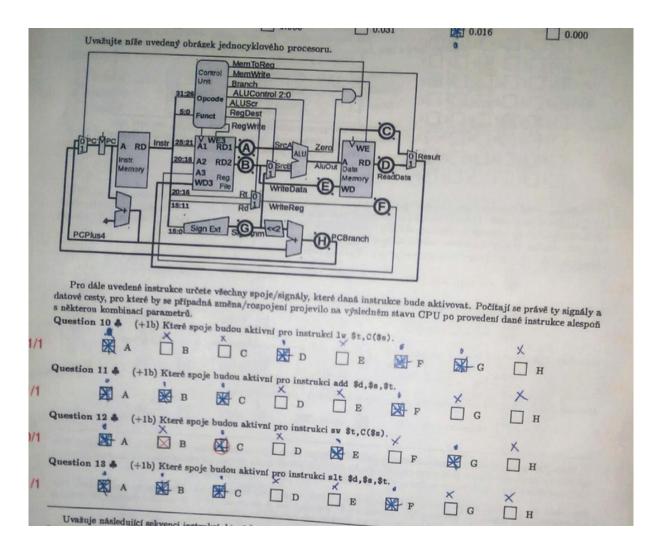












- A vždy
- B ak používame 3 registre, alebo 2 kedy v žiadnom registri nič neprepisujeme (napr. sw)
- C ak nezapisujeme do pamäte ale prepisujeme len registre (add, addi, sub, ..)
- D len pri lw
- E zapis do pamäte(sw)
- F ak prikaz pripisuje register
- G ak sa v príkaze vyskytuje konštanta ©
- H len ak je to jump

instrukci dle	těchto kategorií je násled ce CPU CPI pro třídu	ction) – viz následující t lující: A:20%, B:30%, C	abulku. Dále uvažujt	instrukce v dané kat e program, ve kterém	0.008 egorii dosahuje vždy st zastoupení jednotlivýc
101	12 5	- Pro tridu B	CPI pro třídu C	"DI IVII	
Question 6	(+3b) Jaká je rychlos	procesoru v MIPS (Mi	ilion Instructions Per	1 Same 1)	
383	202	9	□ 81		ý program?
-	že uvedený obrázek zřet	559	392	366	144

Frekvencia je 1Ghz = 1000Mhz pretože CPI je podľa Mhz.

Vzorec pre výpočet je CPI =Frekvencia / (A*CPI_A + B*CPI_B +C*CPI_C + D*CPI_D) 1000/(0.2*5+0.3*5+0.1*5+0.4*1) = 294

Druhy typ úlohy:

1	androty CPI (C)	kterého všechny in cles Per Instruction tegorií je následujíc) – viz následující t	abulku. Dále uvažu	lá instrukce v daně k jte program, ve které	ategorii dosahuje vždy ste- m zastoupení jednotlivých
	Frekvence CPU		CPI pro třídu B	CPI pro třídu C	CPI pro třídu D	
	1GHz	4	1	4	4	
Ques	tion 3 (+1.5)	b) Jak dlouho bude	trvat vykonání pro	ogramu o délce $N =$	109 instrukcí? Odpo	ovídejte v sekundách.
	4.60	□ 6.12	☐ 4.27	3.40	3.61	5.52
12 m	4.60		0.30	6.32	5.83	4.97

CPI po výpočte je 5000/17

Program má 10⁹ inštrukcií.

počet inštrukcií/CPI = 10^9 / (5000/17) = 3400000 po prevode z mikro sekúnd na sekundy je 3,4

strukce není specinkovana nebo za	ce "bne" (v hexadecimálním tvaru) níže uvedeného fragmentu. Pokud některá z položek in- zadání úkolu nevyplývá, pak pro účely tohoto zkouškového testu může daná položka nabývat sak tuto položku nevyplňovat. Operační kód použitých instrukcí je následující: addi:0b001000, rukce bne realizuje operaci: if(\$s != \$t) go to PC+4+4*C.
loop: addi \$17, \$17, -4 sw \$15,-4(\$20) bne \$17, \$3, loop	R opcode(6), 31:26 rs(5), 25:21 rt(5), 20:16 ind(0), 30:26 rs(5), 25:21 rt(5), 20:16 immediate(16), 15:0 J opcode(6), 31:26 address(26), 25:0
Question 1 (+1b) Jak bude instrukce vyjádřené v strojovém Hodnotám 10, 11, 12, 13, 14 a 15	zakódována horní polovina instrukce "bne" v hexadecimálním tvaru? Tzn. horních 16 bitů kódu. (První cifru čísla vyplňte do prvního řádku, druhou cifru do druhého řádku atd. odpovídá A, B, C, D, E a F.
0/1]5
Question 2 (+1b) Jak bude	zakódována dolní polovina instrukce "bne" v hexadecimálním tvaru?
0/1	5

bne \$s \$t C

Opcode rs(17) rt(3) immediate (FFFF – 2 : neviem prečo), 2 pravdepodobne skok dozadu o 2 inštrukcie

0001 01 1000 1 0001 1 FFFC

Uvažujte počítač, ve kterém je nainstalován procesor pracující na konstantní frekvenci 2 0GHz, a který používá tři úrovně cache paměti, kde 16-cestně asociativní L3 cache o velikosti 8MB je sdílená pro instrukce i data a má latenci 12 cyklů. Obě L1 datová a L1 instrukční o velikosti 256 KB používá 4 cesty, je rovněž sdílená pro instrukce i data a má latenci 12 cyklů. Obě L1 datová a L1 instrukční o velikosti 256 KB používá 4 cesty, je rovněž sdílená pro instrukce i data a má latenci 12 cyklů. Obě L1 datová a L1 instrukční o velikosti 256 KB používá 4 cestně asociativní a každá má velikosti 32 KB. Latence obou L1 cache jsou 4 cyklý. Velikost bloku cache je 64 B, cache jsou 4-cestně asociativní o 64 položkách a má latenci 3 cykly. Vyhledání v TLB probíhá paralelně s činnosti L1. Pro danou úroveň cache miss a cache hit trvá stejně dlouho. Celočíselně registry procesoru jsou 64-bitové. Dále je v počítači nainstalováno úroveň cache miss a cache hit trvá stejně dlouho. Celočíselně registry procesoru jsou 64-bitové. Dále je v počítači nainstalováno 16GB hlavní paměti typu DDR4-2400 s časováním CL16-18-18-35, která pracuje na frekvenci 1200 MHz. Pro jednoduchost 16GB hlavní paměti typu DDR4-2400 s časováním CL16-18-18-35, která pracuje na frekvenci 1200 MHz. Pro jednoduchost předpokládejte, že latence paměti je 44 cyklů + 70ns. Dále ve všech případech předpokládejte, že data jsou při čtení z daně úrovně přečtena po uplynutí latence v dané úrovni bez ohledu na objem čtených dat. Sekundární paměť (pevný disk) má úrovně přečtena po uplynutí latence v dané úrovni bez ohledu na objem čtených dat. Sekundární paměť (pevný disk) má kapacitu cca 500.10° B (výrobcem označovanou jako 500GB), ideální (maximální) přenosovou rychlost pro čtení i zápis 540 MB/s a rychlost náhodného čtení/zápisu 97 000 IOPS (Input/Output Operations per Second). Operační systém používá 8 KB stránky. Jedna položka stránkovací tabulky zabírá 8B. Virtuální adresa má šířku 53bitů (počet bítů pro překlad).

F=2GHz

Cache:

Blok =64 B

L3: 8MB, 38 cyklov, 16 cestná

L2: 256KB, 12 cyklov , 4 cestná

L1:32KB, 4 cykly, 4 cestná TLB: 64 položiek, 3 cykly

Ram: 16GB, frekvencia:1200MHz ~ 44cyklov + 70ns

Stránkovanie: 8KB stránky 8B položka

Adresa pre vyber stránky ma šírku 53 bitov

1.Aký je priemerný čas prístupu do pamäti ak miss rate je nasledovný MRL1=0,1, MRL2=0,2, MRL3 = 0,1.

```
Cyklus – čas pre jednu inštrukciu procesora... c = 1/2GHz = 0,5 ns

Cyklus ramky....cr = 1/1,2Ghz = 0.8333 ns

Latence = počet cyklov pre cache* cyklus

t = L1_{atence} + MRL1*(L2_{atence} + MRL2*(L3_{atence} + MRL3*(DDR4_{atence})))

t = 4c + 0.1*(12c + 0.2*(38c + 0.1*(44cr + 70))) = 3,19
```

2. Koľko položiek bude mať stránkovacia tabuľka?

Adresa ma 53 bitov ktorá sa skladá z indexu pre výber stránok a pre umiestnenie v fyzickej adr. $-> 2^{53}$

Jedná stránka ma veľkosť 8 KB = $8192B = 2^{13} \rightarrow 13$ bitov pre kódovanie $2^{53} - 2^{13} = 2^{40}$

3.Koľko bitov potrebujeme pre indexovanie stránkovacej tabuľky?

Jedná stránka ma veľkosť 8 KB = 8191B

8B položka

8192/8 = 1024

 $1024 = 2^{10} \rightarrow 10$ bitov pre kódovanie

4.Ako dlho bude trvať preklade virtuálnej na fyzickú adresu pri page walk ak priemerná latence je 121ns ?

Potrebujeme zistiť počet úrovni stránkovania cez ktoré sa prechádza.

a)

Jedná stránka ma veľkosť 8 KB = 8192B = 2¹³

1024 = 2¹⁰ pre indexovanie stránkovacej tabuľky

$$2^{13} / 2^{10} = 2^3 -> 3$$

Pre indexovanie stránkovacej tabuľky potrebujeme 10 bitov čo je 4 úrovne stránkovania*3 =

12 > 10

b)

2⁴⁰ položiek bude mať stránkovacia tabuľka ak je jednoúrovňová

2¹⁰ pre indexovanie stránkovacej tabuľky

40/10=4 úrovne

4*121ns = 484ns (asi takto)

5.Ako dlho bude trvať preklade virtuálnej na fyzickú, ak sa page walk nevykoná?

Nepotrebujeme stránkovanie , našu adresu našlo TLB.

F=2GHz čo je 2 inštrukcie za 1ns -> 1 inštrukcia za 0,5ns

TLB berie 3 cykly teda 0.5 * 3 = 1.5ns

6.Koľko setov ma L3 cache?

L3: 8MB, 16 cestná Blok =64 B 8MB/(64*16)=8192

7. Hit rate L1 pre 16 MB dokument?

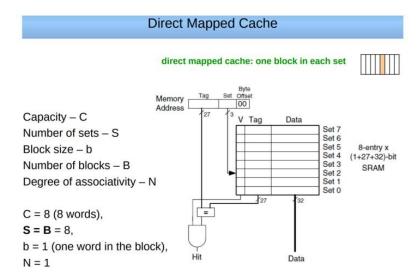
Blok =64 B

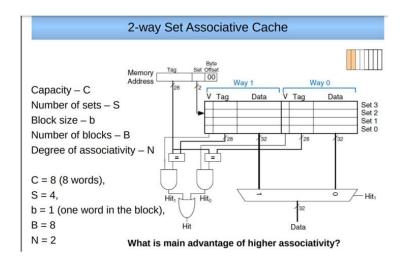
Discord (thx):

7čka mi vyšla jako 1-1/64. Protože procházíš celej ten soubor, tak se data nikdy neopakujou a blok v cache je 64, tak vždycky misneš jenom ten první char a zbytek je hit.

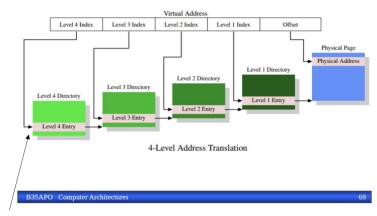
1-1/64=0,98

Kreslenie



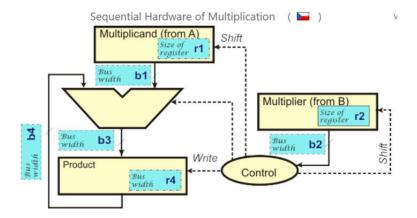


Multi-Levels Page Table

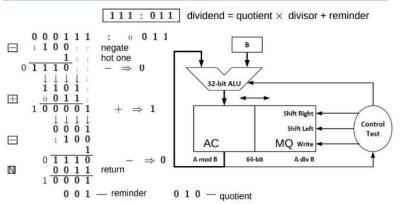


BDPR

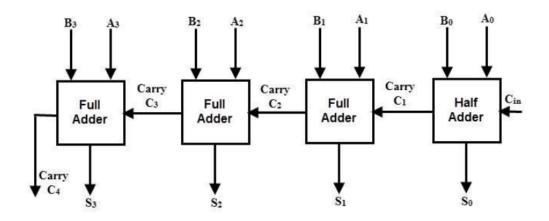
Číslovanie by malo byť od 1 po 4 a nie od 4 po 1, ale postupnosť blokov sedí.



Hardware divider logic (32b case)



Bitova sčitačka:



Dostanete na pisomke

_ !				(1)			ARITI	HMETIC C	ORE INS	TRU	ICTION SE	г	(\circ	OPCODE
and 4) together	MIP 5	Re	fer	ence Data	6		NA	ME, MNEN		FOR MAT		OPERATION	ON		/ FMT /FT / FUNCT (Hex)
ğ ;	CORE INSTRUCT	ION SE	т			OPCODE		On FP Tru			if(FPcond)F				
4			FOR			/ FUNCT	Brancl Divide	On FP Fal	se bclf div	FI R	if(!FPcond) Lo=R[rs]/R			ir(4)	11/8/0/ 0///1a
밀	NAME, MNEMO		MA			(Hex)		Unsigned	divu	R	Lo=R[rs]/R			(6)	
3 a	Add	add	R			0 / 20 _{hex}		d Single	add.s		F[fd]=F[fs		ofverelief	(0)	11/10//0
	Add Immediate	addi	I	R[rt] = R[rs] + SignExtImm	(1,2)	110.00	FP Ad	d	add.d	FR	{F[fd],F[fd-				11/11//0
Fold bottom side (columns	Add Imm. Unsigned			R[rt] = R[rs] + SignExtImm	(2)	110.00	Doubl				ED 1 (1		ft],F[ft+1]}		
<u>=</u>	Add Unsigned	addu	R	R[rd] = R[rs] + R[rt]		0 / 21 _{hex}	FP Co	mpare Sing	le c.x.s*		FPcond = (1 FPcond = (1				11/10//y
8 i	And	and	R	R[rd] = R[rs] & R[rt]		0 / 24 _{hex}	Doubl		c.x.d*	FR	rrcond - ({F[ft],F[ft+	1]; op 1]})?1:0)	11/11//y
) e	And Immediate	andi	I	R[rt] = R[rs] & ZeroExtImm	(3)	c _{hex}					==, <, or <=)	(y is 32, 3	c, or 3e)		
SIC	Branch On Equal	beq	1	if(R[rs]==R[rt])		4 _{hex}		ride Single	div.s	FR	F[fd] = F[fs		1 FFC- : 11)	,	11/10//3
E 1	Dianen on Equal	beq		PC=PC+4+BranchAddr	(4)	nex	FP Div		div.d	FR	{F[fd],F[fd-		s],F[fs+1]} ft],F[ft+1]}		11/11//3
₽ :	Branch On Not Equa	albne	I	if(R[rs]!=R[rt])	(4)	5 _{hex}			e mul.s	FR	F[fd] = F[fs		itj,r[it/1];		11/10//2
<u>ğ</u> I	Jump	_	J	PC=PC+4+BranchAddr PC=JumpAddr	(5)	2 _{hex}	FP Mu		mul.d		{F[fd],F[fd-		s],F[fs+1]}	•	11/11/ /2
<u> </u>		j	-	•			Doubl					{F[1	ft],F[ft+1]}		11/11//2
2 1	Jump And Link	jal	J	R[31]=PC+8;PC=JumpAddr	(5)			otract Single	sub.s	FR	F[fd]=F[fs]				11/10//1
- '	Jump Register	jr	R	PC=R[rs]		0 / 08 _{hex}	FP Sul Doubl		sub.d	FR	{F[fd],F[fd-	+1]} = {F[ti	s],F[fs+1]} ft],F[ft+1]}	-	11/11//1
1	Load Byte Unsigned	lbu	I	$R[rt]={24'b0,M[R[rs] +SignExtImm](7:0)}$	(2)	24 _{hex}		P Single	lwc1	I	F[rt]=M[R[31//
card	Load Halfword			R[rt]={16'b0,M[R[rs]	(2)		Load I	P	ldc1	I	F[rt]=M[R[rs]+SignExt	tImm];	(2)	35//
ទី :	Unsigned	lhu	I	+SignExtImm](15:0)}	(2)	25 _{hex}	Doubl	-			F[rt+1]=M[R[rs]+Sign	ExtImm+4		
perforation to separate	Load Linked	11	I	R[rt] = M[R[rs]+SignExtImm]	(2,7)	$30_{ m hex}$		From Hi	mfhi	R	R[rd] = Hi				0 ///10
l ar	Load Upper Imm.	lui	1	R[rt] = {imm, 16'b0}		fhex		From Lo From Contr	mflo	R R	R[rd] = Lo R[rd] = CR	[re]			0 ///12
e e	Load Word	lw	I	R[rt] = M[R[rs]+SignExtImm]	(2)		Multip		mult	R	$\{Hi,Lo\} = I$		1		0///18
SO	Nor	nor	R	$R[rd] = \sim (R[rs] R[rt])$	(-)	0 / 27 _{hex}	Multip	ly Unsigne	d multu	R	{Hi,Lo} = I			(6)	0///19
<u> </u>	Or	or	R	$R[rd] = R[rs] \mid R[rt]$		0 / 25 _{hex}		light Arith.	sra	R	R[rd] = R[rd]	t] >>> sham	nt		0///3
ੁ ।	Or Immediate	ori	I		(3)			FP Single	swc1	I	M[R[rs]+Si				39//
ᆵ :				R[rt] = R[rs] ZeroExtImm	(3)	THE A	Store I Doubl		sdc1	I	M[R[rs]+Si M[R[rs]+Si			(2)	3d///
은 1	Set Less Than	slt	R	$R[rd] = (R[rs] \le R[rt]) ? 1 : 0$		0 / 2a _{hex}							4] - 1[11.	.,	
<u> </u>	Set Less Than Imm.	slti	I	$R[rt] = (R[rs] \le SignExtImm)$?	1:0(2)	a_{hex}			NT INSTI	RUC	TION FORM	IATS		_	
along j	Set Less Than Imm. Unsigned	sltiu	I	$R[rt] = (R[rs] \le SignExtImm)$? 1:0	(2,6)	b _{hex}	F	R opcod	_	mt	ft	fs	fd		funct
₫ !	Set Less Than Unsig	r eltu	R	$R[rd] = (R[rs] \le R[rt]) ? 1 : 0$		0 / 2b _{hex}	_	31	26 25	_		16 15	11 10	6.5	
<u> </u>	Shift Left Logical	sll	R	$R[rd] = R[rt] \ll shamt$	(0)	0 / 00 _{hex}	F	opeou	26 25	mt	ft 21 20	16 15	immed	liate	
₫ ¦	_							31				16 15			
3 :	Shift Right Logical	srl	R	R[rd] = R[rt] >> shamt		0 / 02 _{hex}	PSEU	DOINSTR	UCTION AME	SET	MNEMON	IIC.	OPERA	TIO	NT.
_	Store Byte	sb	I	M[R[rs]+SignExtImm](7:0) = $R[rt](7:0)$	(2)	28 _{hex}	R	anch Less			blt		<r[rt]) pc<="" td=""><td></td><td></td></r[rt])>		
				M[R[rs]+SignExtImm] = R[rt];	(-)	20		anch Great			bgt		>R[rt]) PC		
2 1	Store Conditional	sc	I	R[rt] = (atomic) ? 1 : 0	(2,7)	$38_{ m hex}$		anch Less			ble		<=R[rt]) P(
Card")	Store Halfword	sh		M[R[rs]+SignExtImm](15:0) =		29 _{hex}		anch Greate ad Immedi		Equ	al bge li		>=R[rt]) P(immediate		.abel
		311	•	R[rt](15:0)	(2)			ove	ate		move	R[rd] =			
("Green	Store Word	sw	I	M[R[rs]+SignExtImm] = R[rt]	(2)		DECI	TED NAM	IE NIIM	BED	USE, CAL				
5 :	Subtract	sub	R	R[rd] = R[rs] - R[rt]	(1)	0 / 22 _{hex}	nEGI:					LCONVE		VED	ACROSS
	Subtract Unsigned	subu	R	R[rd] = R[rs] - R[rt]		0 / 23 _{hex}		NAME N	NUMBER		USE			CAL	
				ise overflow exception			ı	Szero	0	The	Constant Va	lue 0		N.A.	-
Card				Imm = { 16{immediate[15]}, imr	nediate	}	ı	\$at	1	Ass	embler Temp	orary		No	\neg
ر ا				$Imm = \{ 16\{1b'0\}, immediate \}$ $Addr = \{ 14\{immediate[15]\}, imn$	nediate.	2'b0 }		\$v0-\$v1	2-3		ues for Funct			No	
1 #				dr = { PC+4[31:28], address, 2		,	- 1				Expression l	Evaluation			
<u> </u>				ls considered unsigned numbers (-	\$a0-\$a3 \$t0-\$t7	4-7 8-15	_	uments			No No	
ວ '				test&set pair; R[rt] = 1 if pair aton	nic, 0 if	not atomic		\$10-\$17 \$s0-\$s7	16-23		nporaries ed Temporar	iec		Yes	
5 I	BASIC INSTRUCT	ION FO)RM/	ATS			-	St8-St9	24-25		nporaries			No	
Keference Data	R opcode		'S	rt rd shan	nt	funct	ŀ	\$k0-\$k1	26-27		erved for OS	Kernel		No	
. Š	31	26 25		1 20 16 15 11 10	6.5	0	ŀ	Sgp	28		bal Pointer			Yes	$\neg \neg$
	I opcode		'S	rt imme	diate		ŀ	\$sp	29		ck Pointer			Yes	\neg
2		26 25	2	1 20 16 15		0	ı	Sfp	30		me Pointer			Yes	
MIPS	J opcode			address				Sra	31	Ret	urn Address			No	
<u>~</u> '		26 25 Elsevier,	Inc.,	All rights reserved. From Patterso	on and H	dennessy, Co	omputer (Organizatio	n and Des	ign,	4th ed.				

		E CONVER	SION,	ASCII				3	
MIPS	(1) MIPS	(2) MIPS		Deci-	Hexa-	ASCII	Deci-	Hexa-	ASCI
opcode	funct	funct	Binary	mal		Char-	mal	deci-	Char-
31:26)	(5:0)	(5:0)			mal	acter	mai	mal	acter
(1)	sll	add.f	00 0000	0	0	NUL	64	40	(a)
		sub.f	00 0001	1	1	SOH	65	41	A
j	srl	mul.f	00 0010		2	STX	66	42	В
jal	sra	div.f	00 0011	3	3	ETX	67	43	C
ped	sllv	sqrt.f	00 0100		4	EOT	68	44	D
bne		abs.f	00 0101	5	5	ENQ	69	45	E
blez	srlv	mov.f	00 0110		6	ACK	70	46	F
bgtz	srav	neg.f	00 0111	7	7	BEL	71	47	G
addi	jr		00 1000		8	BS	72	48	Н
addiu	jalr		00 1001	9	9	HT	73	49	I
slti	movz		00 1010		a	LF	74	4a	J
sltiu	movn		00 1011	11	b	VT	75	4b	K
andi	syscall	round.w.f	00 1100		c	FF	76	4c	L
ori	break	trunc.w.f	00 1101	13	d	CR	77	4d	M
xori		ceil.wf	00 1110		e	SO	78	4e	N
lui	sync	floor.w.f		15	f	SI	79 80	4f 50	O P
(70)	mfhi		01 0000		10	DLE			
(2)	mthi mflo		01 0001	17 18	11 12	DC1 DC2	81 82	51 52	Q R
		movz.f	01 0010	18	13	DC2 DC3	82	53	S
	mtlo	movn.f	01 0100		14	DC4	84	54	T
			01 0100	21	15	NAK	85	55	Ü
			01 0101		16	SYN	86	56	v
			01 0111	23	17	ETB	87	57	w
	mult		01 1000		18	CAN	88	58	X
	multu		01 1000	25	19	EM	89	59	Ŷ
	div		01 1001		la	SUB	90	5a	ž
	divu		01 1010	27	1b	ESC	91	5b	1
	0170		01 1100		le	FS	92	5c	_
			01 1101	29	ld	GS	93	5d	ì
			01 1110		le	RS	94	5e	
			01 1111	31	1f	US	95	5f	
lb	add	cvt.s.f	10 0000	32	20	Space	96	60	-
1h	addu	cvt.d.f	10 0001	33	21	!	97	61	a
lwl	sub		10 0010	34	22		98	62	b
lw	subu		10 0011	35	23	#	99	63	c
lbu	and	cvt.w.f	10 0100	36	24	Ş	100	64	d
1hu	or		10 0101	37	25	%	101	65	e
lwr	xor		10 0110	38	26	&	102	66	f
	nor		10 0111	39	27		103	67	g
sb			10 1000	40	28	(104	68	h
sh			10 1001	41	29)	105	69	i
swl	slt		10 1010		2a		106	6a	j
SW	sltu		10 1011	43	2b	+	107	6b	k
			10 1100		2c	,	108	6c	- 1
			10 1101	45	2d	-	109	6d	m
SWI			10 1110		2e		110	6e	n
cache			10 1111	47	2f	/	111	6f	0
11	tge	c.f.f	11 0000	48	30	0	112	70	p
lwc1	tgeu	c.un,f	11 0001	49	31	1	113	71	q
lwc2	tlt	c.eq.f	11 0010		32	2	114	72	Г
pref	tltu	c.ueq.f	11 0011	51	33	3	115	73	S
	teq	c.olt.f	11 0100		34	4	116	74	t
ldc1		c.ult.f	11 0101	53	35	5	117	75	u
1dc2	tne	c.ole.f	11 0110		36	6	118	76	v
		c.ule.f	11 0111	55	37	7	119	77	W
sc		c.sf.f	11 1000		38	8	120	78	X
swc1		c.ngle.f	11 1001	57	39	9	121	79	y
swc2		c.seq.f	11 1010		3a	:	122	7a	Z
		c.ngl.f	11 1011	59	3b	- ;	123	7b	-{
		c.lt.f	11 1100		3c	<	124	7e	Į
sdc1		c.nge.f	11 1101	61	3d		125	7d	}
sdc2		c.le.f	11 1110	62	3e	> 2	126	7e	DEI

c.1ef | 11 110 | 62 | 8e | 126 | 7e | 7e | 111 | 111 | 63 | 3f | 7e | 127 | 7f | DEL | (1) opcode(31:26) == 0 (2) opcode(31:26) == $17_{\text{ten}} (11_{\text{hex}})$; if fmt(25:21)== $16_{\text{ten}} (10_{\text{hex}})$ f = s (single); if fmt(25:21)== $17_{\text{ten}} (11_{\text{hex}}) f = d \text{ (double)}$

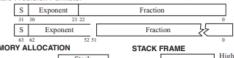
Copyright 2009 by Elsevier, Inc., All rights reserved. From Patterson and Hennessy, Comp.

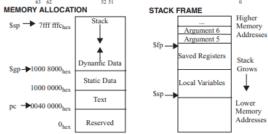
IEEE 754 FLOATING-POINT STANDARD

 $(-1)^S \times (1 + Fraction) \times 2^{(Exponent - Bias)}$ where Single Precision Bias = 127, Double Precision Bias = 1023.

IEEE Single Precision and Double Precision Formats:







DATA ALIGNMENT

Double Word								
	Wo	rd		Word				
Halfword		Halfword		Halfword		Halfword		
Byte	Byte	Byte	Byte	Byte	Byte	Byte	Byte	
0	Ι	2	3	4	5	6	7	

Value of three least significant bits of byte address (Big Endian)

EXCEPTION CONTROL REGISTERS: CAUSE AND STATUS

B D	Interru Mask		Exception Code	
31	15	8	6	2
	Pendin	g	U	EI
	Interru	pt	M	L E
	15	- 8	4	1 0

BD = Branch Delay, UM = User Mode, EL = Exception Level, IE =Interrupt Enable

Ε,	KCEPIIC	ON CC	DES			
	Number	Name	Cause of Exception	Number	Name	Cause of Exception
	0	Int	Interrupt (hardware)	9	Bp	Breakpoint Exception
	4	AdEL	Address Error Exception	10	RI	Reserved Instruction
	-	Auce	(load or instruction fetch)	10	KI	Exception
	- 5	AdES	Address Error Exception	11	CpU	Coprocessor
		Aues	(store)	- 11	СрО	Unimplemented
	6	IBE	Bus Error on	12	Ov	Arithmetic Overflow
			Instruction Fetch	12	Ov	Exception
	7	DBE	Bus Error on	13	Tr	Trap
	_ ′	DOE	Load or Store	1.3		*
	8	Sys	Syscall Exception	15	FPE	Floating Point Exception

SI Size	Prefix	Symbol	IEC Size	Prefix	Symbol
10 ³	Kilo-	K	210	Kibi-	Ki
10^{6}	Mega-	M	220	Mebi-	Mi
10 ⁹	Giga-	G	230	Gibi-	Gi
10^{12}	Tera-	T	240	Tebi-	Ti
10 ¹⁵	Peta-	P	250	Pebi-	Pi
10^{18}	Exa-	E	260	Exbi-	Ei
10^{21}	Zetta-	Z	270	Zebi-	Zi
1024	Yotta-	Y	280	Yobi-	Yi