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:

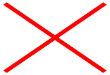
1. Napřed vyjádříme celé číslo binárně. 510 = 101b
2. vezmeme si desetinnou část čísla - 0.71. Desetinnou část budeme násobit dvěma a to tolikrát, kolik bitů má naše mantisa - tedy v tomto případě pětkrát. Z výsledku oddělíme celou část, to je náš výsledek hodnoty bitu a s desetinnou částí opakujeme proces.

0.71 × 2 = 1. 420.42 × 2 = 0.820.84 × 2 = 1.680.68 × 2 = 1.360.36 × 2 = 0.72

1. Nyní dáme obě části dohromady

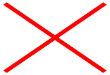
101.10110

1. Desetinnou tečku posuneme o tolik pozic doleva, aby před desetinnou tečkou byla jen jedna jednička. V našem případě o 2 pozice.



1. 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 = 24-1 - 1 = 8 - 1 = 7, exponent = 2 + 7 = 9, binárně 1001.

1. 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).

1. Znaménkový bit - 0, pokud je číslo kladné, 1, pokud je číslo záporné.

1. Výsledek:

5.71 = 0100101101

0 - znaménko

1001 - exponent

01101 - mantisa

1. **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

1. **Jaká bude dekadická hodnota tohoto největšího čísla?**

Převedeme z binárky zpět:

1. Určíme hodnotu exponentu

11102 = 1410 , od tohoto čísla odečteme aditivní konstantu, 14 - 7 = 7. Čili náš exponent bude 27.

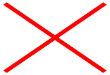
1. Určíme hodnotu mantisy

Sčítáme postupně

2-1 + 2-2 + 2-3 + 2-4 + 2-5 = 31/32 (v našem případě až do 2-5 , protože máme 5 bitů pro mantisu)

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

2^7 \* 63/32

252

(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)

1. **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

1. **Jaká bude hodnota tohoto nejmenšího čísla?**

Převedeme bity pro exponent:

0001b = 110, odečteme aditivní konstantu 1-7 = -6.

Exponent = 2-6

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

1. **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

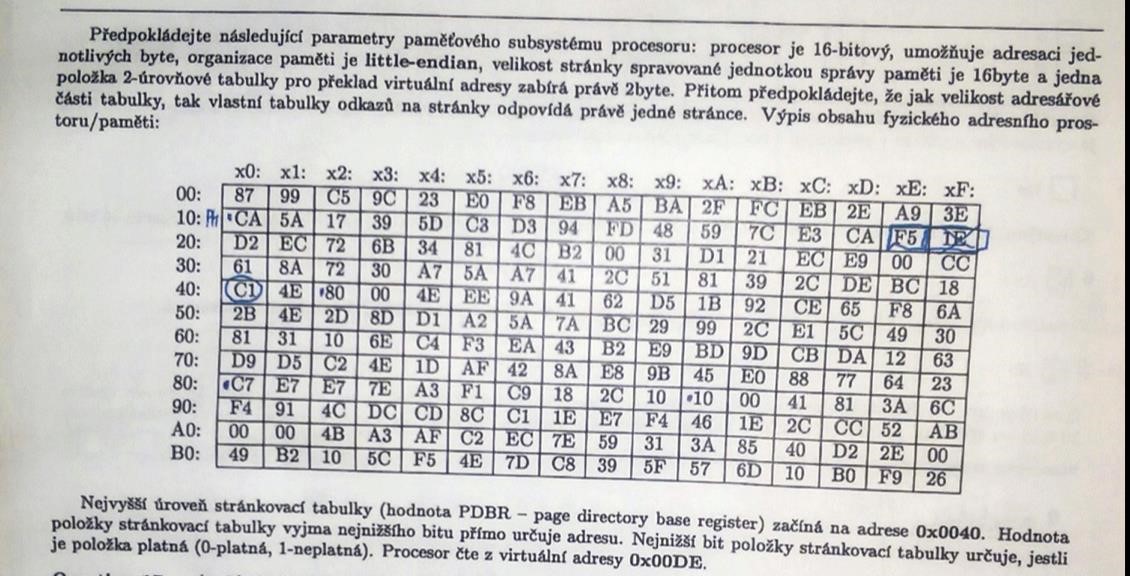
# STRÁNKOVÁNÍ

Ze zadání úlohy potřebuji vyčíst:

-počet úrovní

-velikost stránky

-kolik bytes zabírá jedna položka



**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 = 24

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

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

velikost stránky / velikost jedné položky

16 / 2 = 8 = 23

index stránky je tedy určen třemi bity

1. kolik mám úrovní, tolikrát vynásobím bity pro index

2 úrovně - 3 + 3 bitů pro indexy

1. výsledek

4 + 3 + 3 = 10

S uvedenými parametry je schopna přeložit 10 bitů.

1. **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

1. **Jaký je index v tabulce stránek první (a druhé) úrovně?**

Virtuální adresu ze zadání si přepíšu do binární soustavy

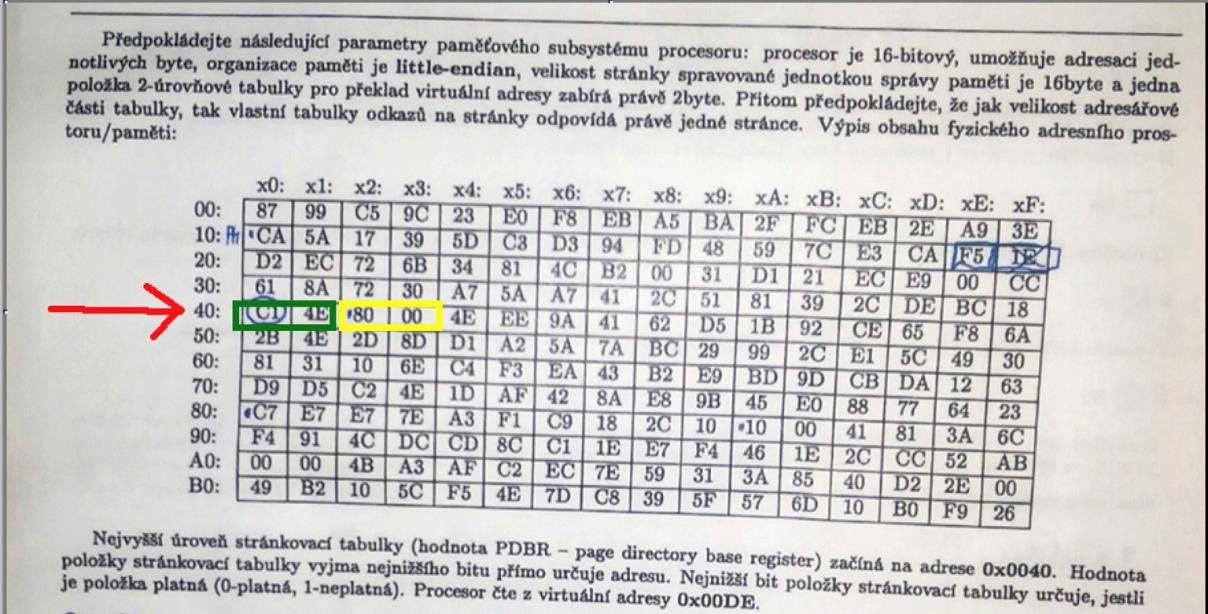
0x00DE = 0000 0000 1101 1110 index je určen třemi bity

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

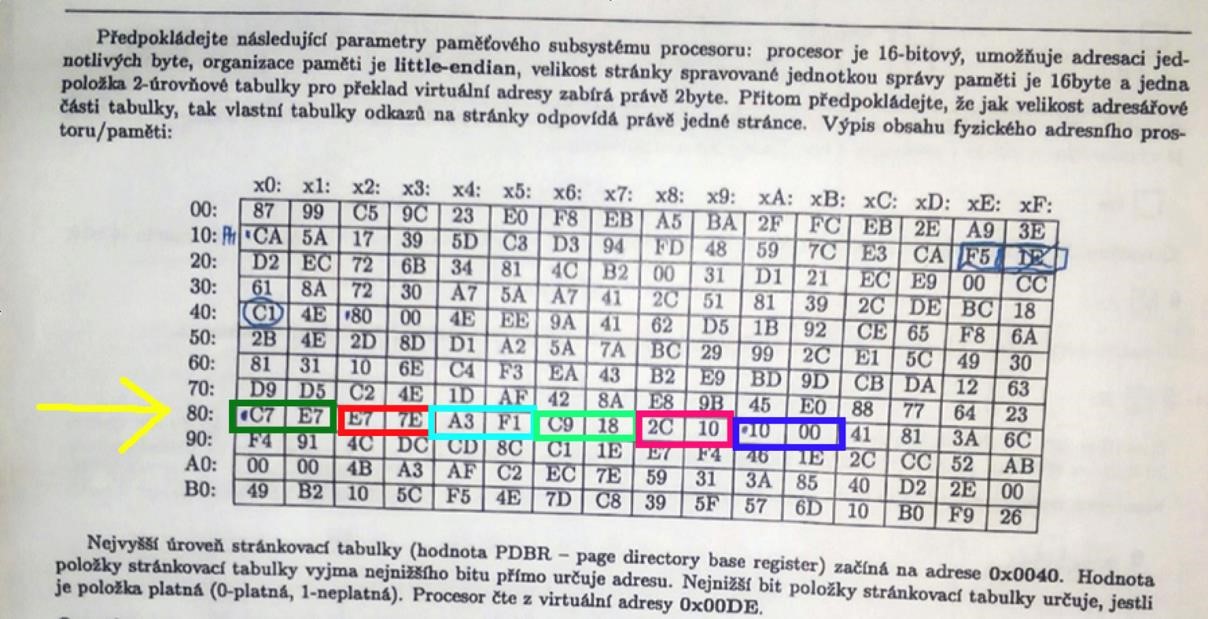
0000 0000 1101 1110 červená - offset (předchozí otázka) modrá - index 2. úrovně zelená - index 1. úrovně převedu opět do dec:

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

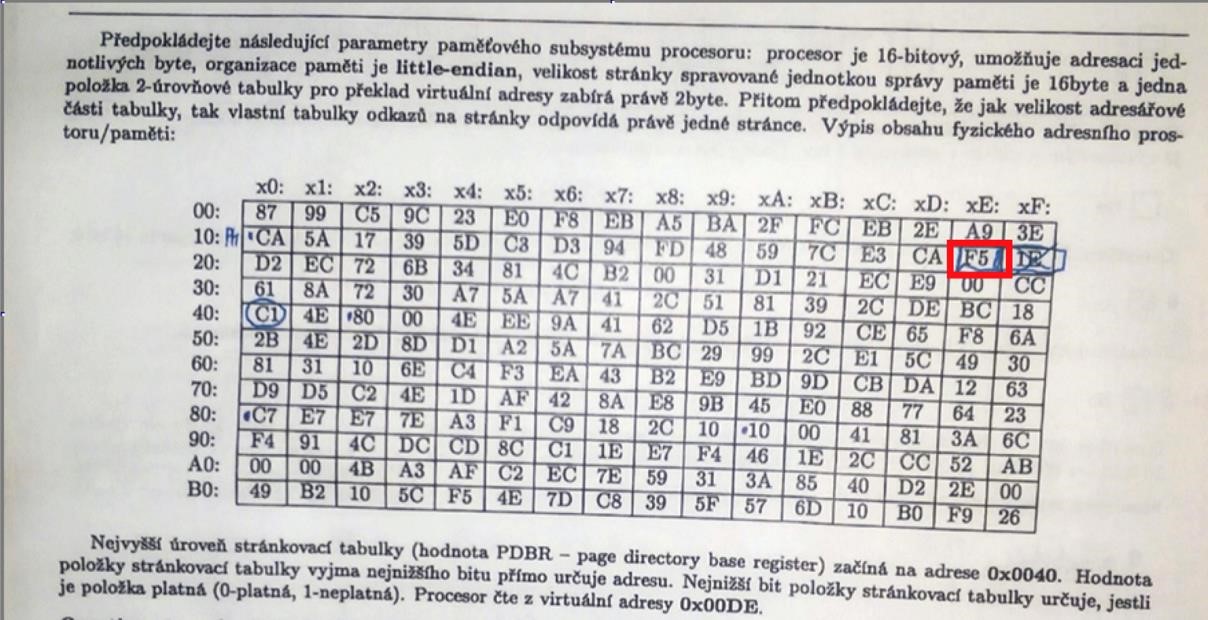
1. **Jaká data budou z adresy načtena?**
2. vezmu startovací adresu (v zadání) - 0x0040
3. posouvám se vpravo o index 1. úrovně. Pozor, posouvám se po dvojících, jelikož velikost položky je zrovna 2 B. Začínám na indexu 0.

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

1. 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

 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:



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.

1. Tato moje v zadání překládá 10 bitů a má dvě úrovně.

1. Přidáním další úrovně budu schopná přeložit 13 bitů.

Ještě další 16 bitů. A to hledám. Musím proto přidat 2 další úrovně.

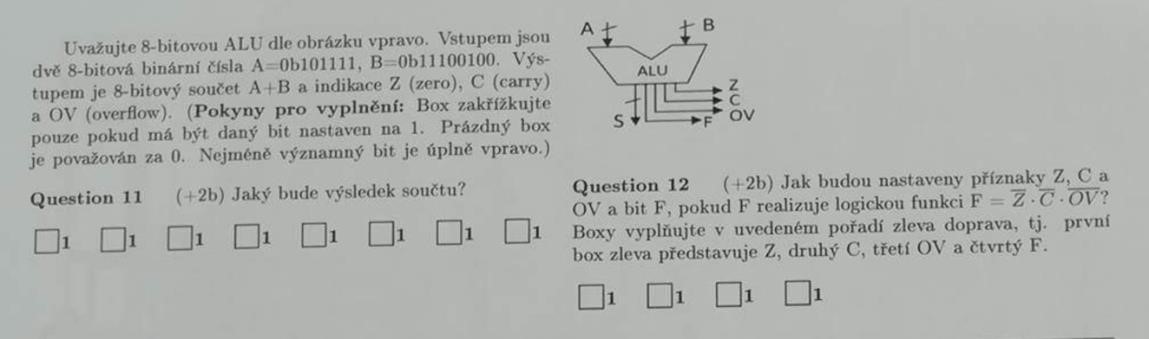
1. Celkem tedy potřebuji mít 4 úrovně, aby byl pokrytý překlad z 16-bit virt. na 16-bit fyz.

(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



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

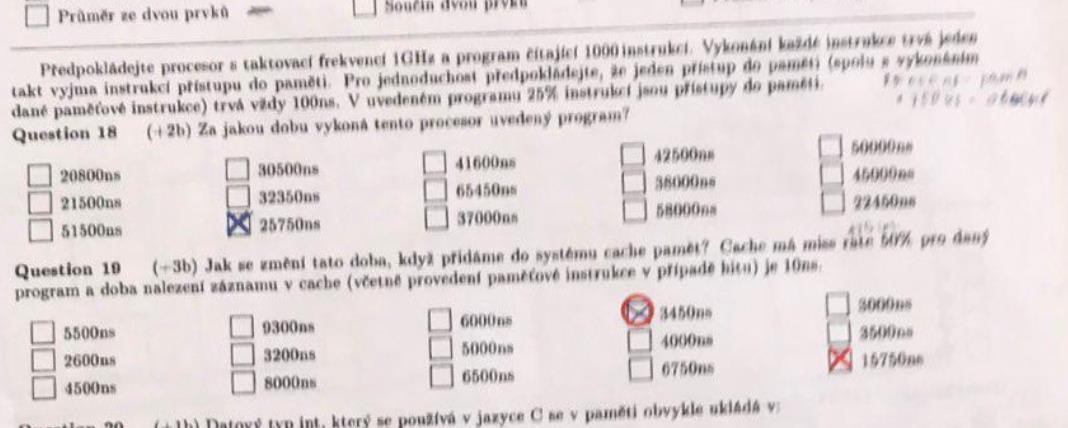
F ... nastavený, pokud jsou všechny předchozí příznaky na nule, tedy negace logického součinu Z, C a OVs

[http://teaching.idallen.com/dat2343/10f/notes/040\_overflow.txt](http://teaching.idallen.com/dat2343/10f/notes/040_overflow.txt?fbclid=IwAR1ysQM5Rzs1lX6i-aORbqs7MVqLnWbAtWgcg0_0vNXrBn0YdXQpHMpRGSU)

# 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.

 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 × 100 + 750 × 1 = 25750𝑛𝑠

1. **Jak se změní tato doba, když přidáme do systému cache paměť?**

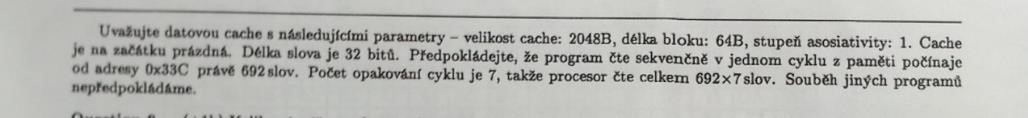
Ze zadání vzčteme miss rate a dobu nalezení záznamu cache.

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 × 10 + 750 × 1 + 125 × 100 = 15750𝑛𝑠

# CACHE



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

1. **Kolik má cache setů?**

Spočítám podle vzorce

= = 32 setov

1. **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:

12 % 32 = 12

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:

3        3         C

         0011  0011  1100

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 → 691 / 16 = 43,1875→  44x budeme načítat do cache

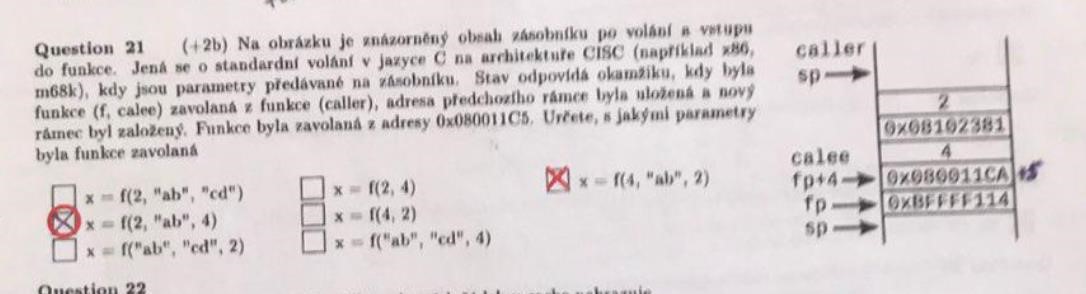
Tedy 1 + 44 missů a 692 – 45 = 647 hitů

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

# ARGUMENTY FUNKCE



Všimněme si v obrázku, kam směřují popisky.

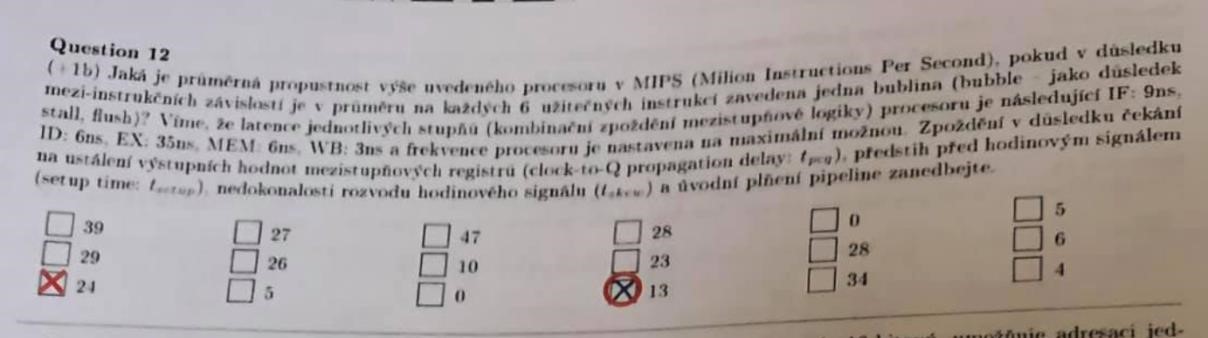
Argumenty hledáme mezi řádky "calee" a "sp".

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

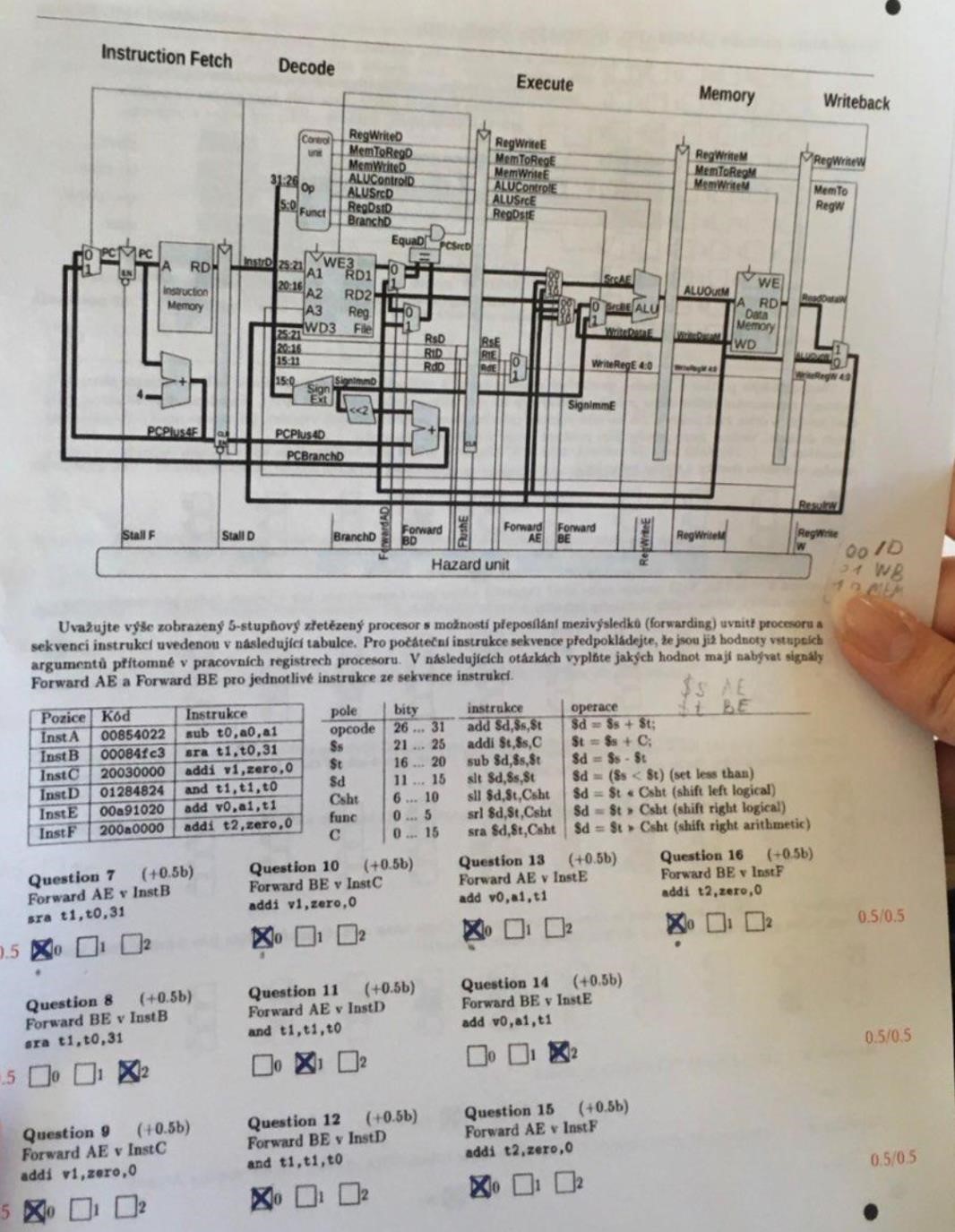


Zjistím poměr užitečných instrukcí / bublina ze zadání: každá sedmá „instrukce“ je bublina tedy poměr je 6/7 Spočítám frekvenci procesoru: najdu, která fáze má nejvyšší latenci, vezmu hodnotu této latence a spočítám, kolikrát se vleze do 1 s vyjde mi frekvence v Hz, tu převedu do MHz a vynásobím poměrem

1 s / 35 ns = 28,6 MHz

28,57 MHz x 6/7 = 24.48 = 24

# FORWARDING

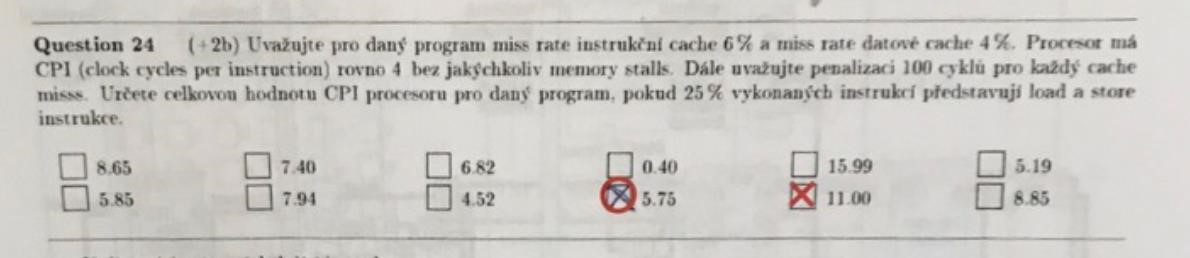


Máme dva signály - AE a BE. signál AE je pro registr $s, signál BE je pro registr $t (viz instrukce vpravo vedle kódu) Závislost je hned pod sebou - signál 2, závislost je ob jeden řádek - signál 1, mezera je víc než jeden řádek - signál 0.

Pokud má instrukce konstantu a daný registr ($s nebo $t) nepoužívá, dáme do signálu 0.

V obrázku jsou všechny hodnoty správné.

# URČENÍ HODNOTY CPI



Ze zadání zjistíme:

default CPI = 4 miss rate instr. cache = 6 % miss rate dat. cache = 4 %

za cache miss penalizace = 100 cyklů

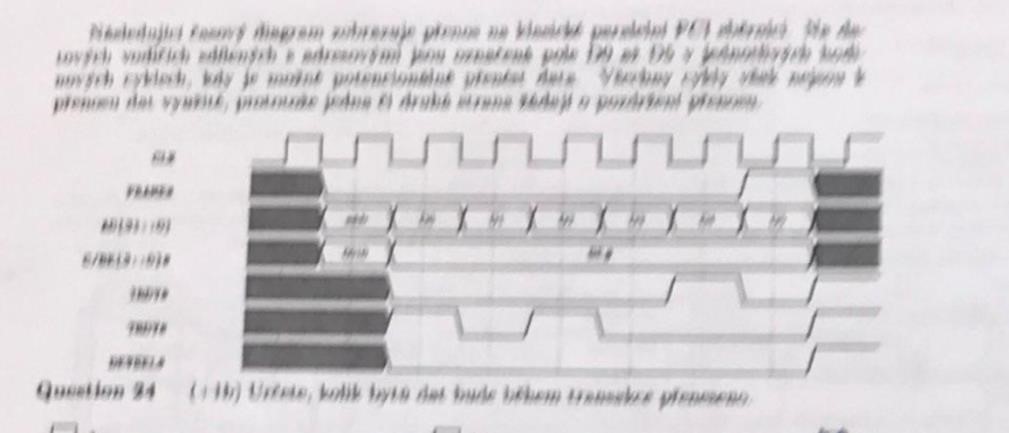
25 % instrukcí jsou load a store

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 x 1 x 100) + (0.04 x 0.25 x 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 *\\_(*ツ*)\_/¯*

3 x 4 = 12 -bude přeneseno 12 Bytů

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

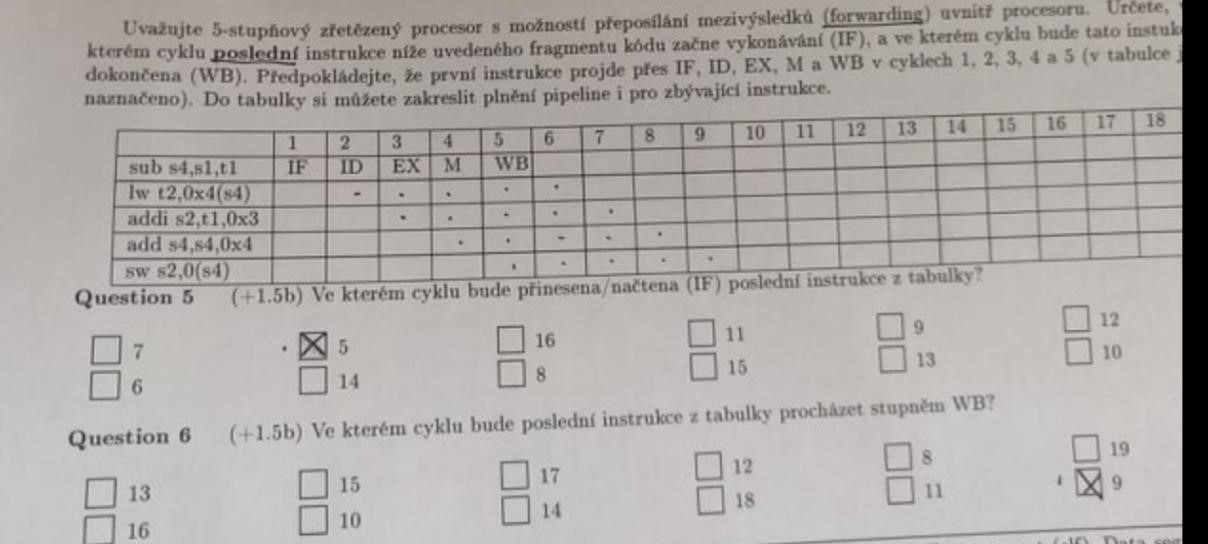
ANO/NE

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

 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

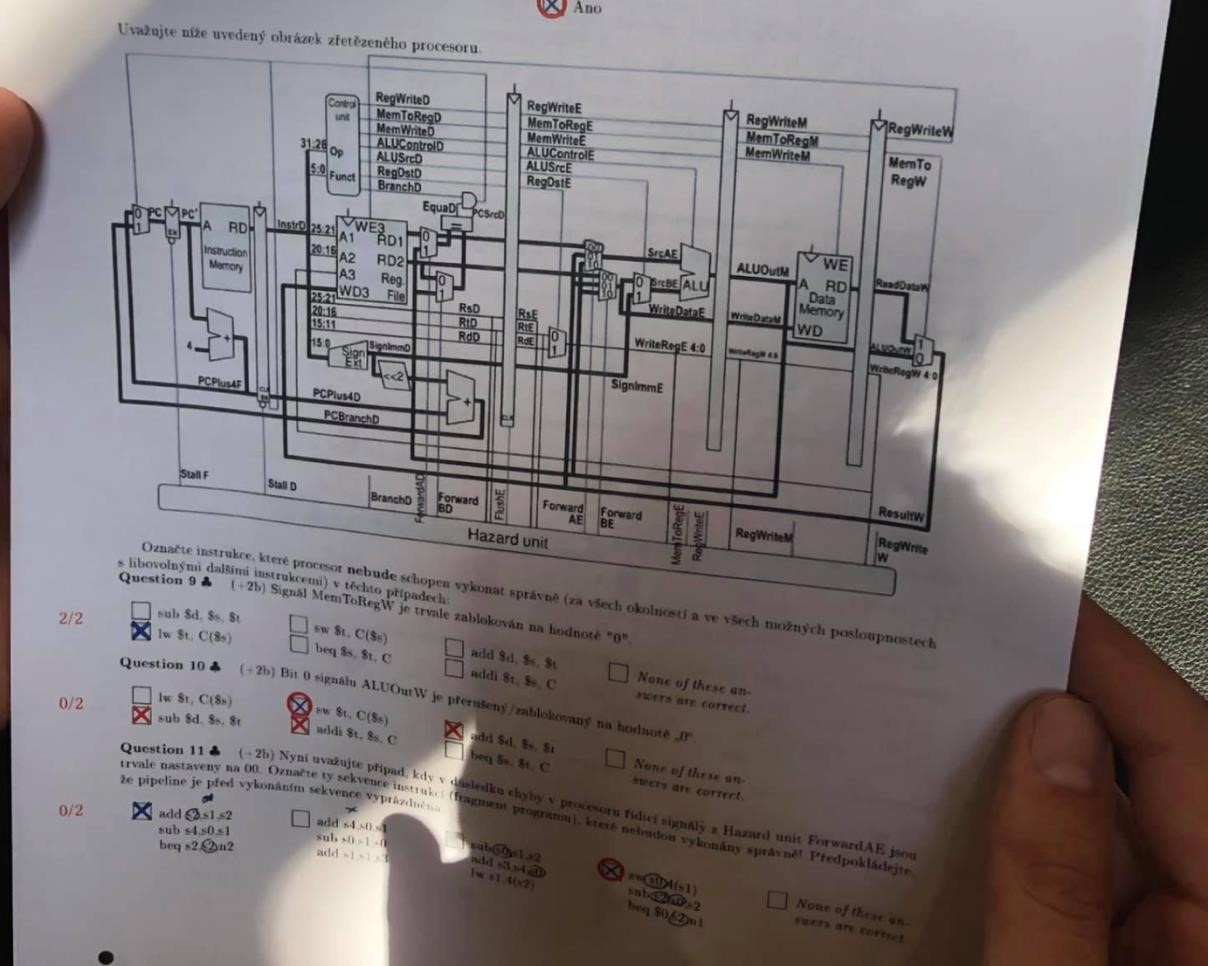
-mezi aritmetickými instrukcemi forwarding vyřeší všechno, nemusím dávat žádné stally

-mezi aritmetickou a paměťovou (lw, sw) musím dát jeden stall, pokud je tam závislost

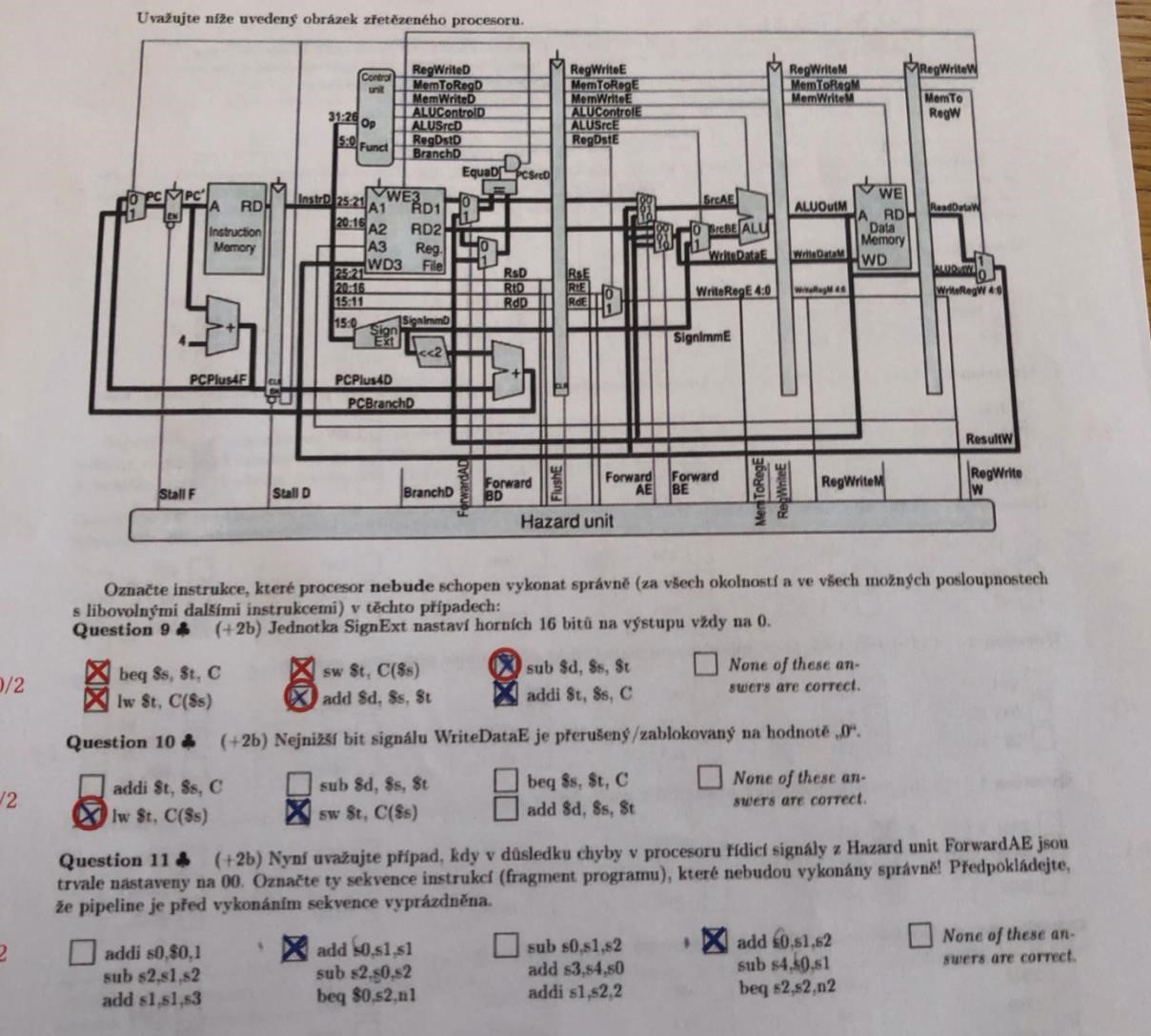
-branch instrukce: pokud instrukce skoku zavisi na instrukce před ní, musí nastat stall, protože branch instrukce se vyhodnocuje ve fázi ID a není možné přeposílat z fáze EX do ID

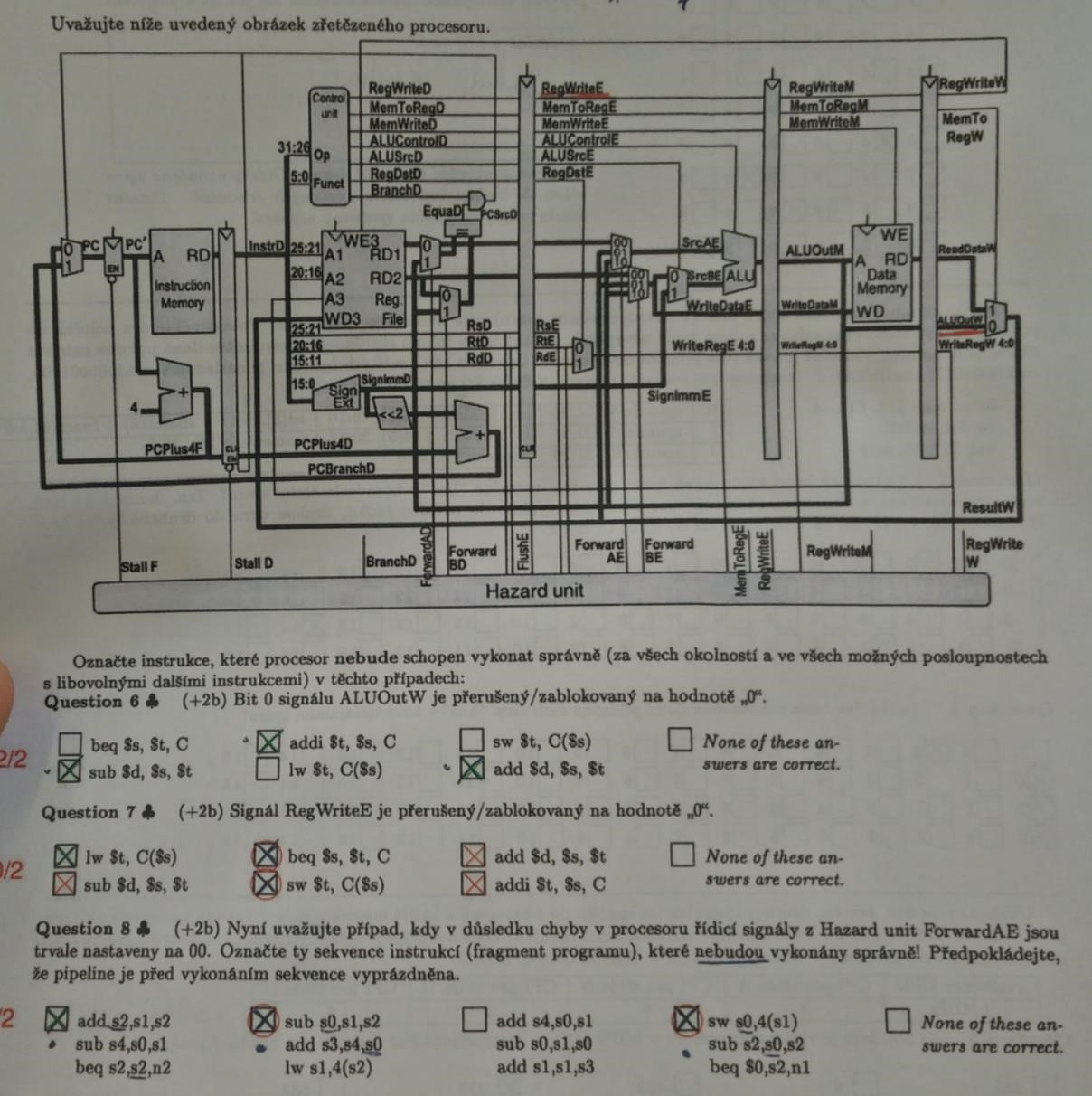
btw. není to sice opravené, ale odpovědi jsou správné, můžete si to ověřit pomocí QtMipsu

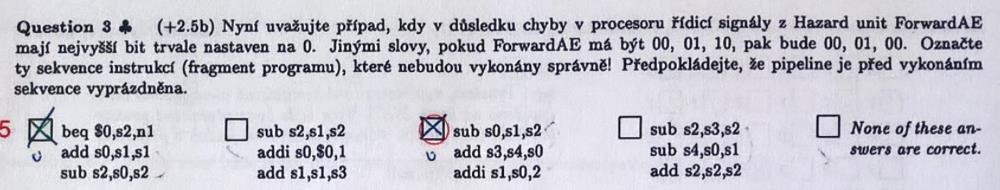
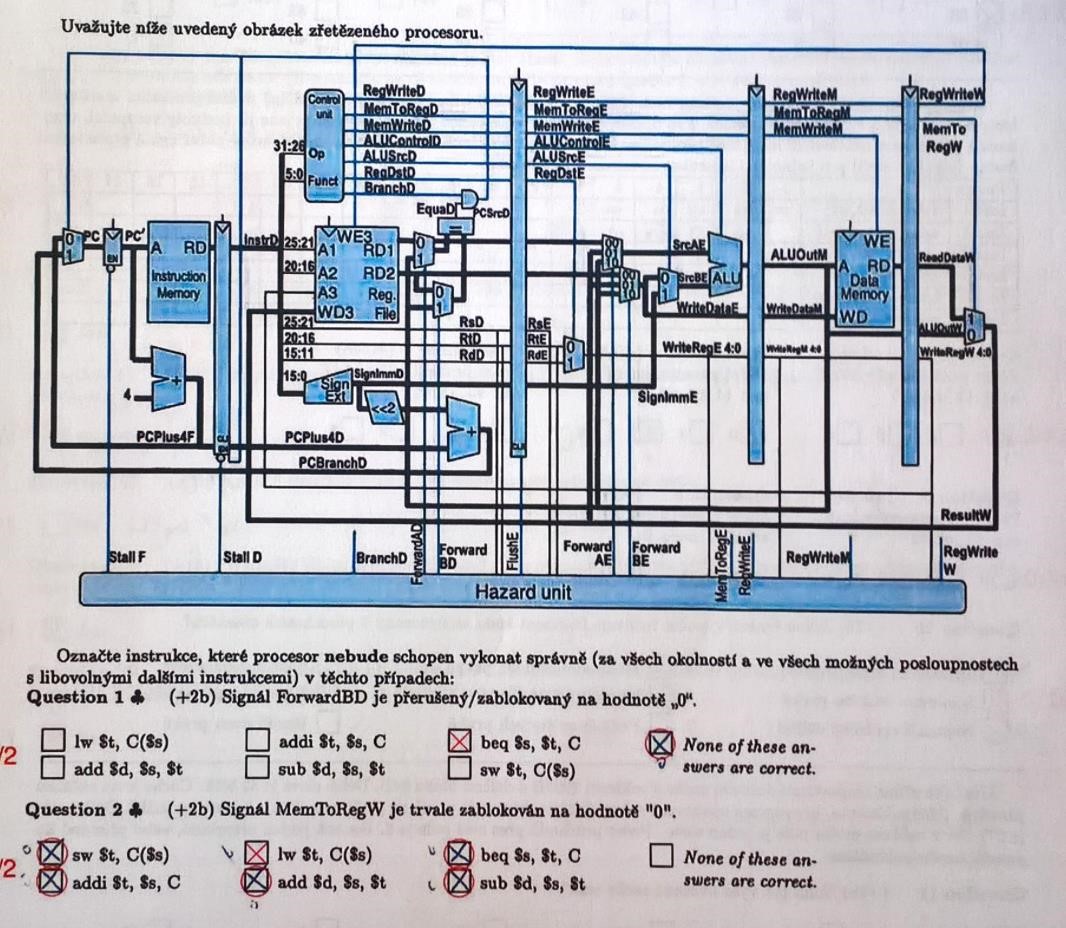
# BETTER LEARN THIS

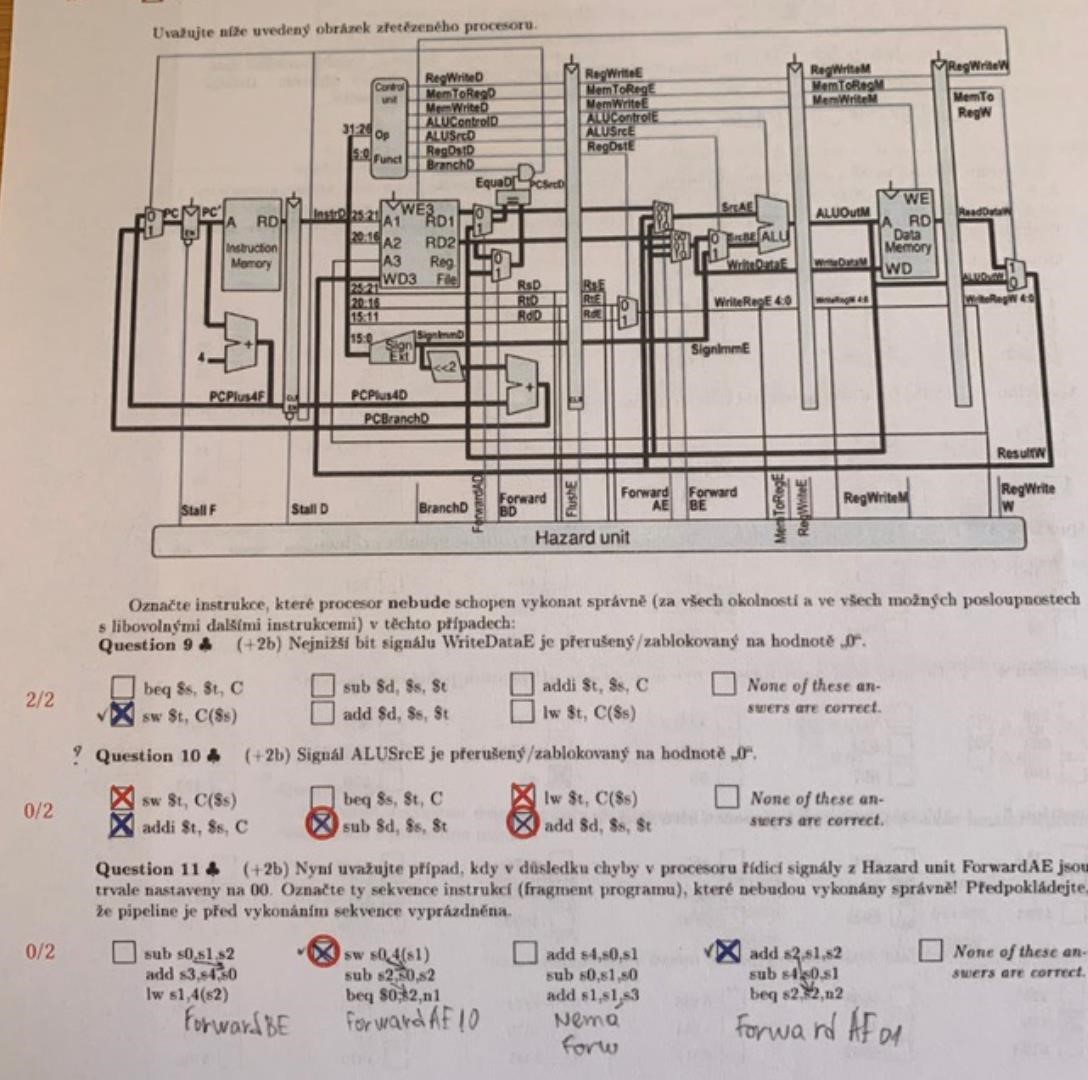


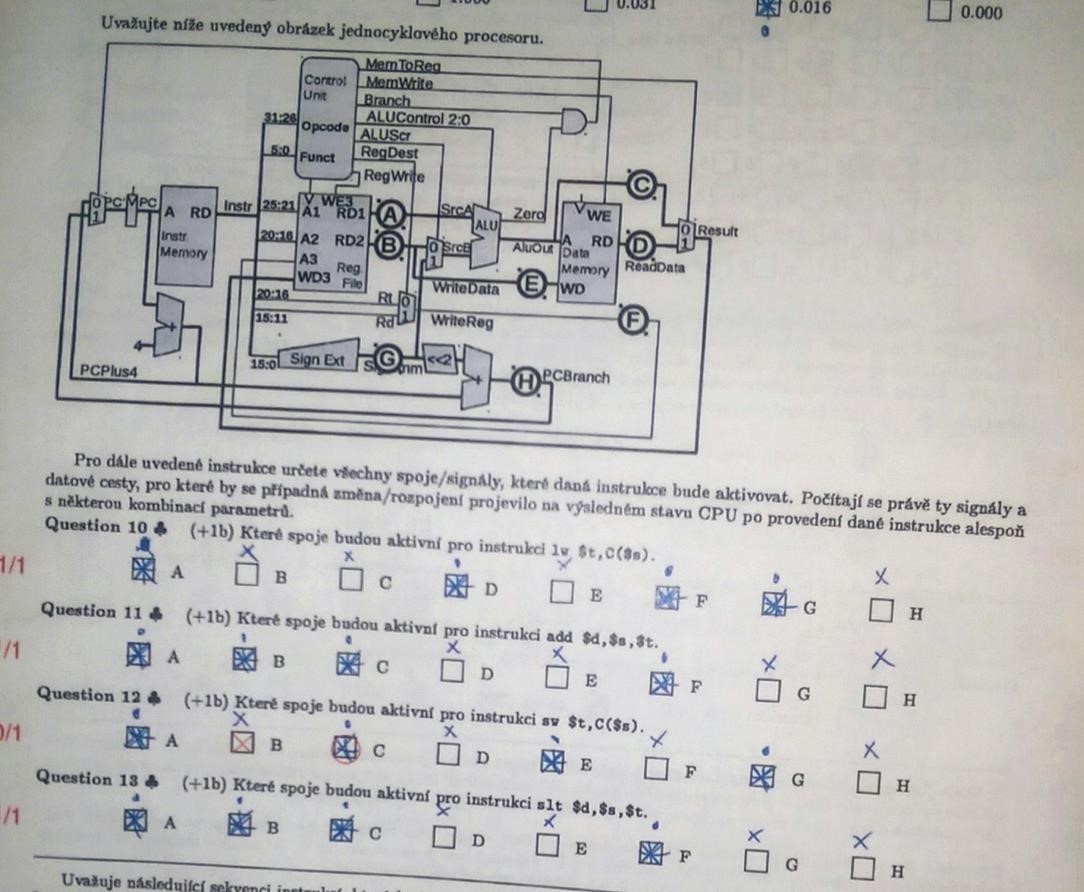
-











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, ..)

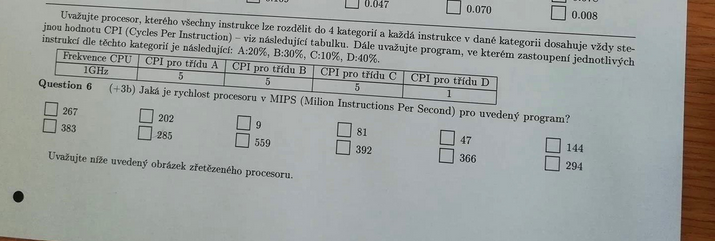
D - len pri lw

E - zapis do pamäte(sw)

F - ak prikaz pripisuje register

G - ak sa v príkaze vyskytuje konštanta ©

H - len ak je to jump

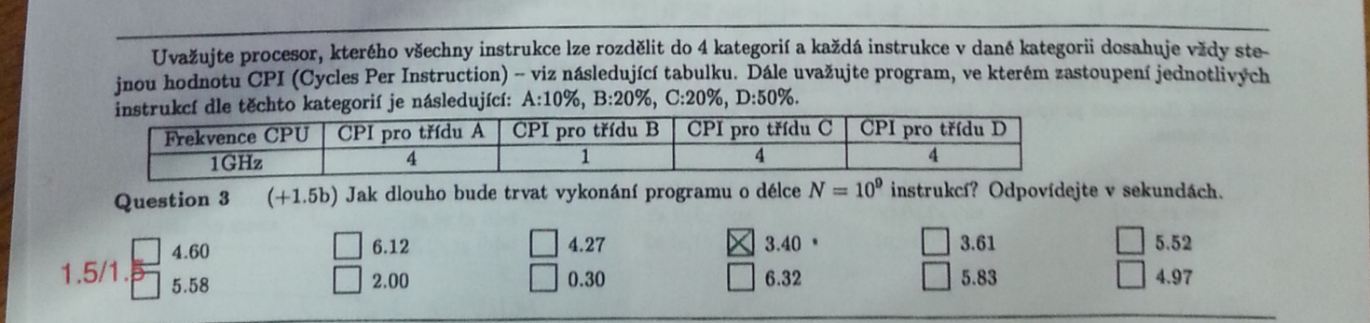


Frekvencia je 1Ghz = 1000Mhz pretože CPI je podľa Mhz.

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

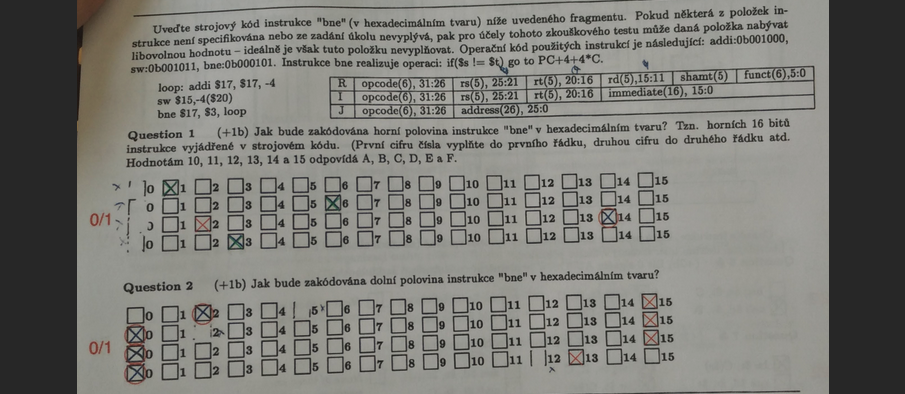
Druhy typ úlohy:



CPI po výpočte je 5000/17

Program má 109 inštrukcií.

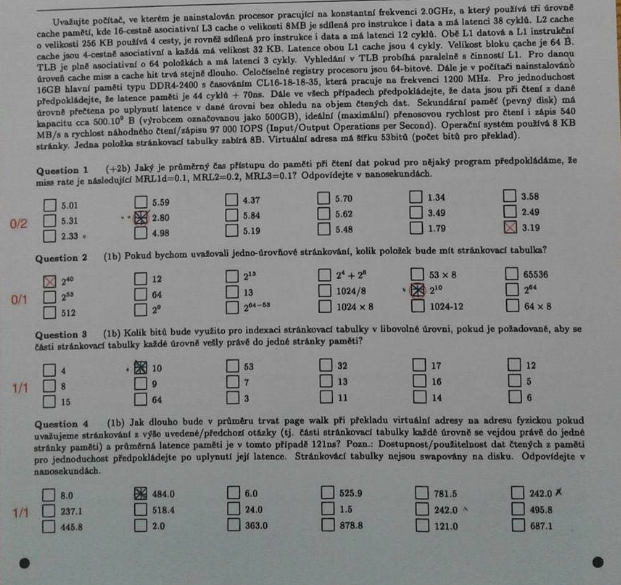
počet inštrukcií/CPI = 109 / (5000/17) = 3400000 po prevode z mikro sekúnd na sekundy je 3,4



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



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\_latence + MRL1 \*(L2\_latence+ MRL2 \*(L3\_latence + MRL3 \*(DDR4\_latence)))

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. -> 253

Jedná stránka ma veľkosť 8 KB = 8192B = 213 -> 13 bitov pre kódovanie

253 - 213 = 240

**3.Koľko bitov potrebujeme pre indexovanie stránkovacej tabuľky?**

Jedná stránka ma veľkosť 8 KB = 8191B

8B položka

8192/8 = 1024

1024 = 210 -> 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

213 / 210 =23 -> 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?**

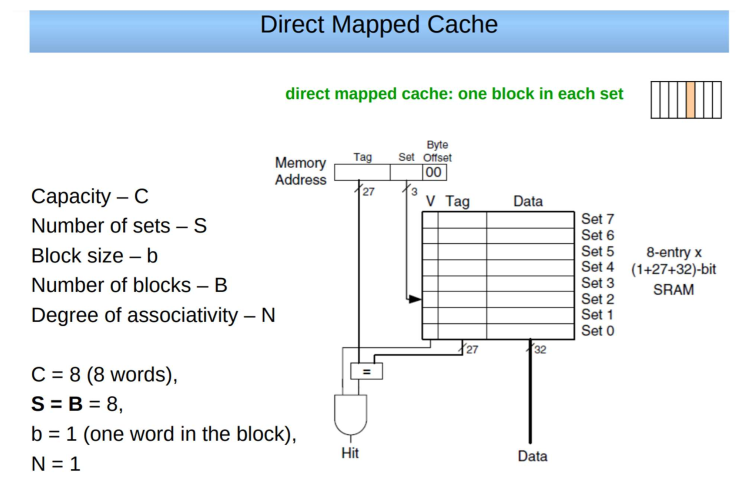
Blok =64 B

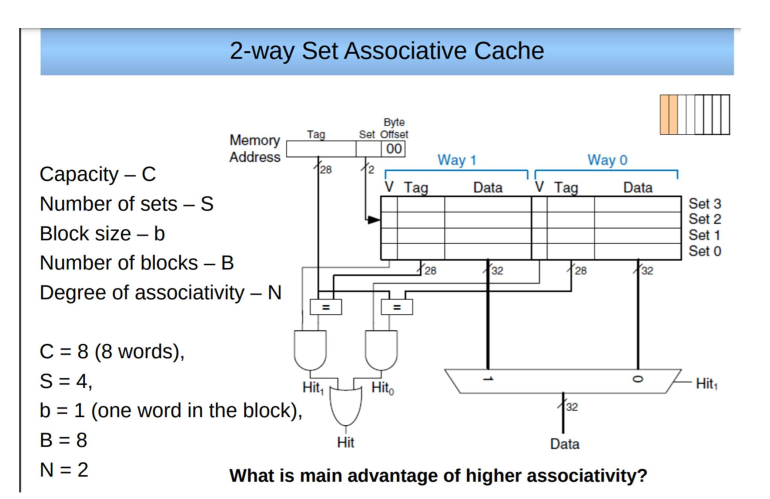
Discord (thx):

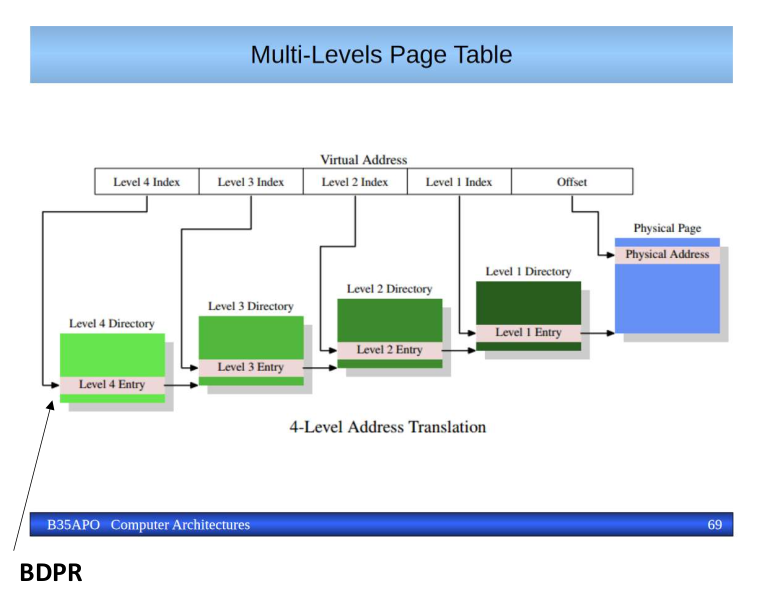


1-1/64=0,98

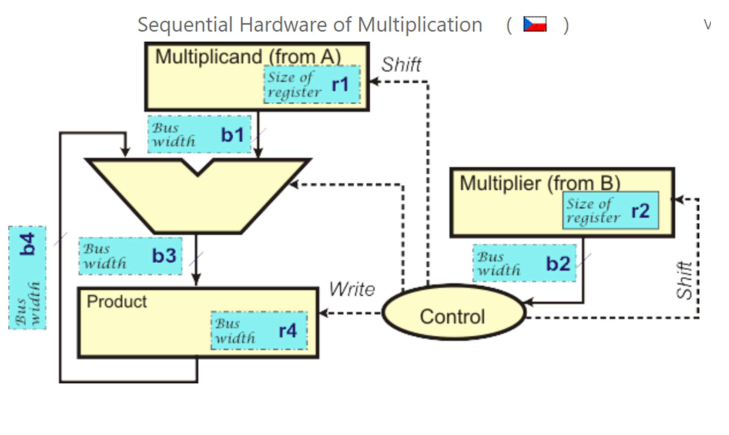
**Kreslenie**

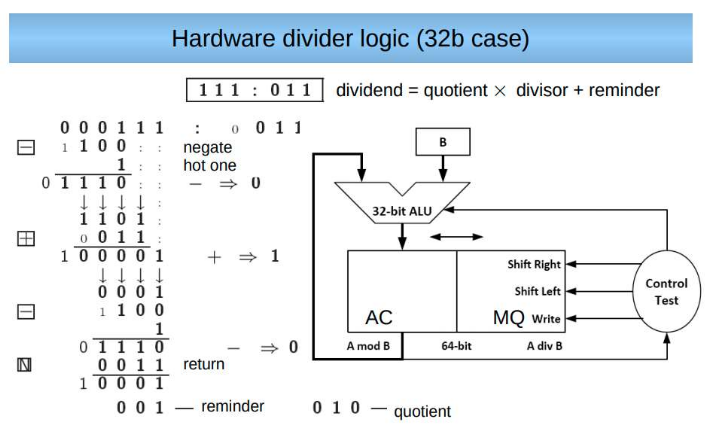




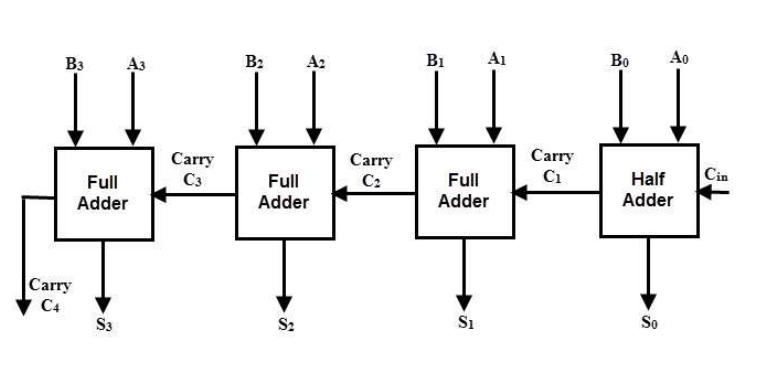


Číslovanie by malo byť od 1 po 4 a nie od 4 po 1, ale postupnosť blokov sedí.





**Bitova sčitačka:**



**Dostanete na pisomke**

