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. $5_{10} = 101_b$
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.

$$0.71 \times 2 = 1.42$$
$$0.42 \times 2 = 0.82$$
$$0.84 \times 2 = 1.68$$
$$0.68 \times 2 = 1.36$$
$$0.36 \times 2 = 0.72$$

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.

$$1.0110110$$

5. Zjistím hodnotu exponentu
Exponent se rovná počtu pozic, o kolik jsme v minulém kroku posouvali + aditivní konstanta.
Aditivní konstanta: $k = 2^{a-1} - 1$, 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
$2^{-1} + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-5} = 31/32$ (v našem případě až do 2^{-5}, protože máme 5 bitů pro mantisu)
3. Obě čísla vynásobíme
$2^7 \times \frac{31}{32} = 2^2 \times 31 = 124$
4. Přičteme 128,
výsledek 124 + 128 = 252.0

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

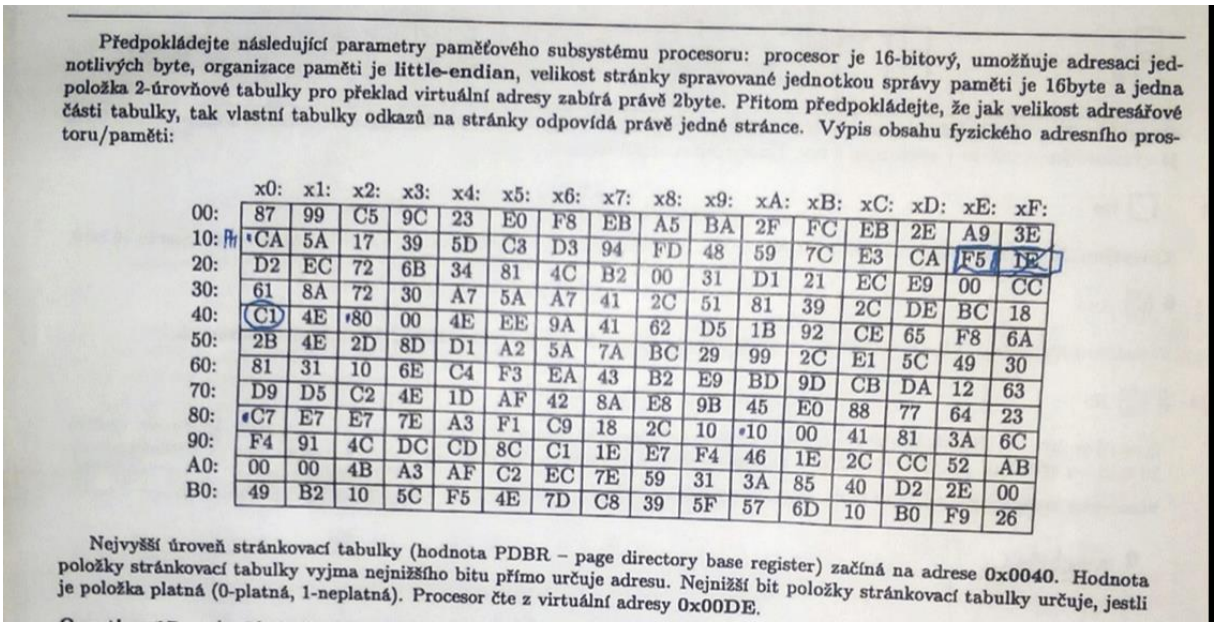
0000100000

5) Jaká bude hodnota tohoto nejmenšího čísla?

Převedeme bity pro exponent:
$0001_b = 1_{10}$,
odečteme aditivní konstantu 1-7 = -6.
Exponent = 2^{-6}
Hodnota nejmenšího čísla je 2^{-6} = 0.015625, zaokrouhlíme 0.016

STRÁNKOVÁNÍ

Ze zadání úlohy potřebuji vyčíst:
-počet úrovní
-velikost stránky
-kolik bytes zabírá jedna položka



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 prostoru/paměti:

	x0:	x1:	x2:	x3:	x4:	x5:	x6:	x7:	x8:	x9:	xA:	xB:	xC:	xD:	xE:	xF:
00:	87	99	C5	9C	23	E0	F8	EB	A5	BA	2F	FC	EB	2E	A9	3E
10:	CA	5A	17	39	5D	C3	D3	94	FD	48	59	7C	E3	CA	F5	DE
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC	E9	00	CC
30:	61	8A	72	30	A7	5A	A7	41	2C	51	81	39	2C	DE	BC	18
40:	C1	4E	80	00	4E	EE	9A	41	62	D5	1B	92	CE	65	F8	6A
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C	E1	5C	49	30
60:	81	31	10	6E	C4	F3	EA	43	B2	E9	BD	9D	CB	DA	12	63
70:	D9	D5	C2	4E	1D	AF	42	8A	E8	9B	45	E0	88	77	64	23
80:	C7	E7	E7	7E	A3	F1	C9	18	2C	10	10	00	41	81	3A	6C
90:	F4	91	4C	DC	CD	8C	C1	1E	E7	F4	46	1E	2C	CC	52	AB
A0:	00	00	4B	A3	AF	C2	EC	7E	59	31	3A	85	40	D2	2E	00
B0:	49	B2	10	5C	F5	4E	7D	C8	39	5F	57	6D	10	B0	F9	26

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 je 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 2^n
   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 2^n
   velikost stránky / velikost jedné položky
   16 / 2 = 8 = 2^3
   index stránky je tedy určen třemi bity
3. kolik mám úrovní, tolikrát vynásobím bity pro index
   2 úrovně - 3 + 3 bitů pro indexy
4. výsledek

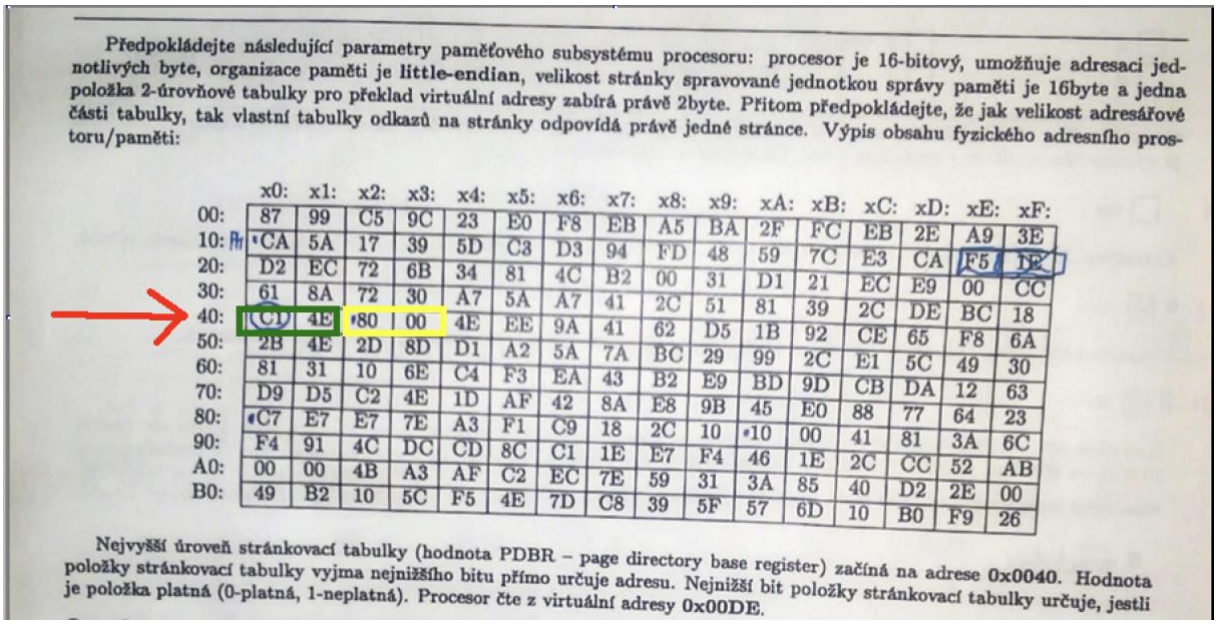
4 + 3 + 3 = 10
S uvedenými parametry je schopna přeložit 10 bitů.

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.

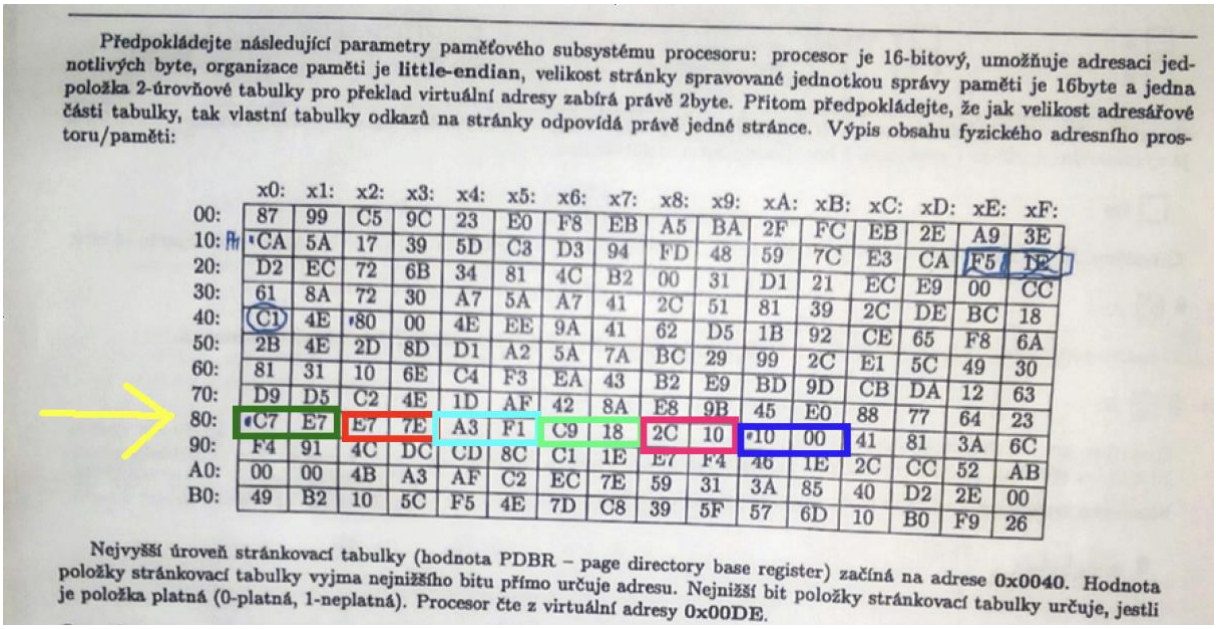


Předpokládejte následující parametry paměťového subsystému procesoru: procesor je 16-bitový, umožňuje adresaci jednotlivých byte, organizace paměti je little-endian, velikost stránky spravované jednotkou správy paměti je 16byte a jedna položka 2-úrovňové tabulky pro překlad virtuální adresy zabírá právě 2byte. Přitom předpokládejte, že jak velikost adresářové části tabulky, tak vlastní tabulky odkazů na stránky odpovídá právě jedné stránce. Výpis obsahu fyzického adresního prostoru/paměti:

	x0:	x1:	x2:	x3:	x4:	x5:	x6:	x7:	x8:	x9:	xA:	xB:	xC:	xD:	xE:	xF:
00:	87	99	C5	9C	23	E0	F8	EB	A5	BA	2F	FC	EB	2E	A9	3E
10:	CA	5A	17	39	5D	C3	D3	94	FD	48	59	7C	E3	CA	F5	DE
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC	E9	00	CC
30:	61	8A	72	30	A7	5A	A7	41	2C	51	81	39	2C	DE	BC	18
40:	CD	4E	80	00	4E	EE	9A	41	62	D5	1B	92	CE	65	F8	6A
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C	E1	5C	49	30
60:	81	31	10	6E	C4	F3	EA	43	B2	E9	BD	9D	CB	DA	12	63
70:	D9	D5	C2	4E	1D	AF	42	8A	E8	9B	45	E0	88	77	64	23
80:	C7	E7	E7	7E	A3	F1	C9	18	2C	10	10	00	41	81	3A	6C
90:	F4	91	4C	DC	CD	8C	C1	1E	E7	F4	46	1E	2C	CC	52	AB
A0:	00	00	4B	A3	AF	C2	EC	7E	59	31	3A	85	40	D2	2E	00
B0:	49	B2	10	5C	F5	4E	7D	C8	39	5F	57	6D	10	B0	F9	26

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 je položka platná (0-platná, 1-neplatná). Procesor čte z virtuální adresy 0x00DE.

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

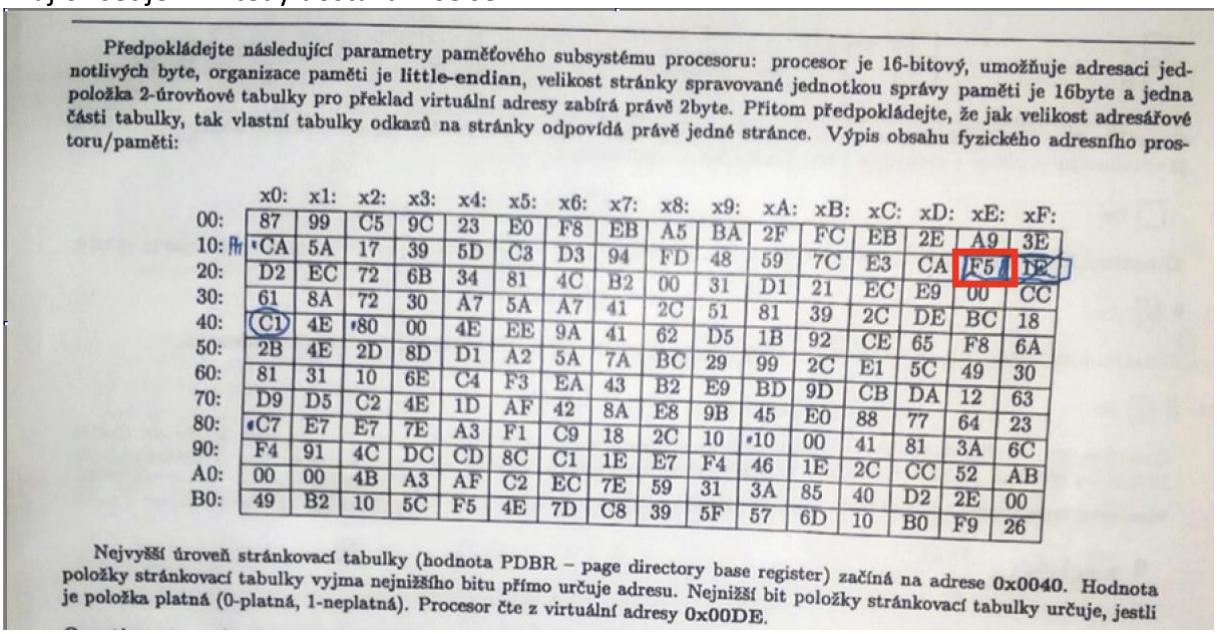


Předpokládejte následující parametry paměťového subsystému procesoru: procesor je 16-bitový, umožňuje adresaci jednotlivých byte, organizace paměti je little-endian, velikost stránky spravované jednotkou správy paměti je 16byte a jedna položka 2-úrovňové tabulky pro překlad virtuální adresy zabírá právě 2byte. Přitom předpokládejte, že jak velikost adresářové části tabulky, tak vlastní tabulky odkazů na stránky odpovídá právě jedné stránce. Výpis obsahu fyzického adresního prostoru/paměti:

	x0:	x1:	x2:	x3:	x4:	x5:	x6:	x7:	x8:	x9:	xA:	xB:	xC:	xD:	xE:	xF:
00:	87	99	C5	9C	23	E0	F8	EB	A5	BA	2F	FC	EB	2E	A9	3E
10:	CA	5A	17	39	5D	C3	D3	94	FD	48	59	7C	E3	CA	F5	DE
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC	E9	00	CC
30:	61	8A	72	30	A7	5A	A7	41	2C	51	81	39	2C	DE	BC	18
40:	C1	4E	80	00	4E	EE	9A	41	62	D5	1B	92	CE	65	F8	6A
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C	E1	5C	49	30
60:	81	31	10	6E	C4	F3	EA	43	B2	E9	BD	9D	CB	DA	12	63
70:	D9	D5	C2	4E	1D	AF	42	8A	E8	9B	45	E0	88	77	64	23
80:	C7	E7	E7	7E	A3	F1	C9	18	2C	10	10	00	41	81	3A	6C
90:	F4	91	4C	DC	CD	8C	C1	1E	E7	F4	46	1E	2C	CC	52	AB
A0:	00	00	4B	A3	AF	C2	EC	7E	59	31	3A	85	40	D2	2E	00
B0:	49	B2	10	5C	F5	4E	7D	C8	39	5F	57	6D	10	B0	F9	26

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 je položka platná (0-platná, 1-neplatná). Procesor čte z virtuální adresy 0x00DE.

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 sem:



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 prostoru/paměti:

	x0:	x1:	x2:	x3:	x4:	x5:	x6:	x7:	x8:	x9:	xA:	xB:	xC:	xD:	xE:	xF:
00:	87	99	C5	9C	23	E0	F8	EB	A5	BA	2F	FC	EB	2E	A9	3E
10:	CA	5A	17	39	5D	C3	D3	94	FD	48	59	7C	E3	CA	F5	DE
20:	D2	EC	72	6B	34	81	4C	B2	00	31	D1	21	EC	E9	00	CC
30:	61	8A	72	30	A7	5A	A7	41	2C	51	81	39	2C	DE	BC	18
40:	C1	4E	80	00	4E	EE	9A	41	62	D5	1B	92	CE	65	F8	6A
50:	2B	4E	2D	8D	D1	A2	5A	7A	BC	29	99	2C	E1	5C	49	30
60:	81	31	10	6E	C4	F3	EA	43	B2	E9	BD	9D	CB	DA	12	63
70:	D9	D5	C2	4E	1D	AF	42	8A	E8	9B	45	E0	88	77	64	23
80:	C7	E7	E7	7E	A3	F1	C9	18	2C	10	10	00	41	81	3A	6C
90:	F4	91	4C	DC	CD	8C	C1	1E	E7	F4	46	1E	2C	CC	52	AB
A0:	00	00	4B	A3	AF	C2	EC	7E	59	31	3A	85	40	D2	2E	00
B0:	49	B2	10	5C	F5	4E	7D	C8	39	5F	57	6D	10	B0	F9	26

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 je položka platná (0-platná, 1-neplatná). Procesor čte z virtuální adresy 0x00DE.

Odpověď: Z adresy budou načtena data F5.

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.

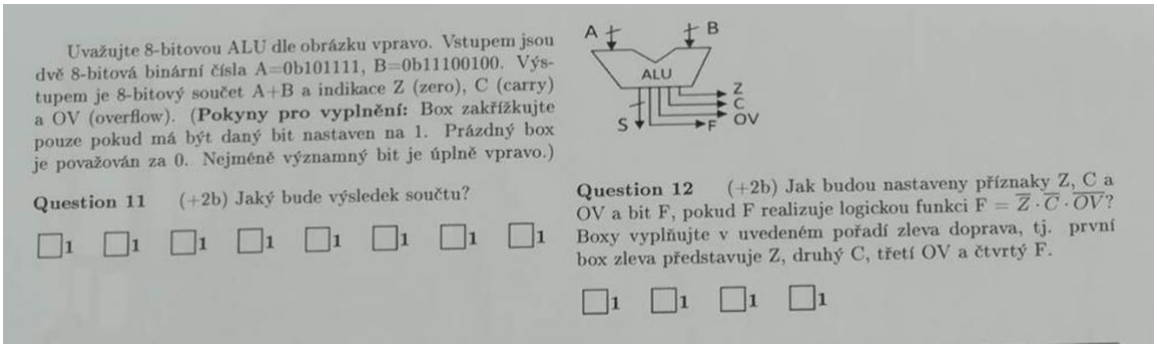
3. Přidáním další úrovně budu schopná přeložit 13 bitů.
Ještě další 16 bitů. A to hledám. Musím proto přidat 2 další úrovně.

4. Celkem tedy 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


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.)

Question 11 (+2b) Jaký bude výsledek součtu?

☐1 ☐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.

☐1 ☐1 ☐1 ☐1

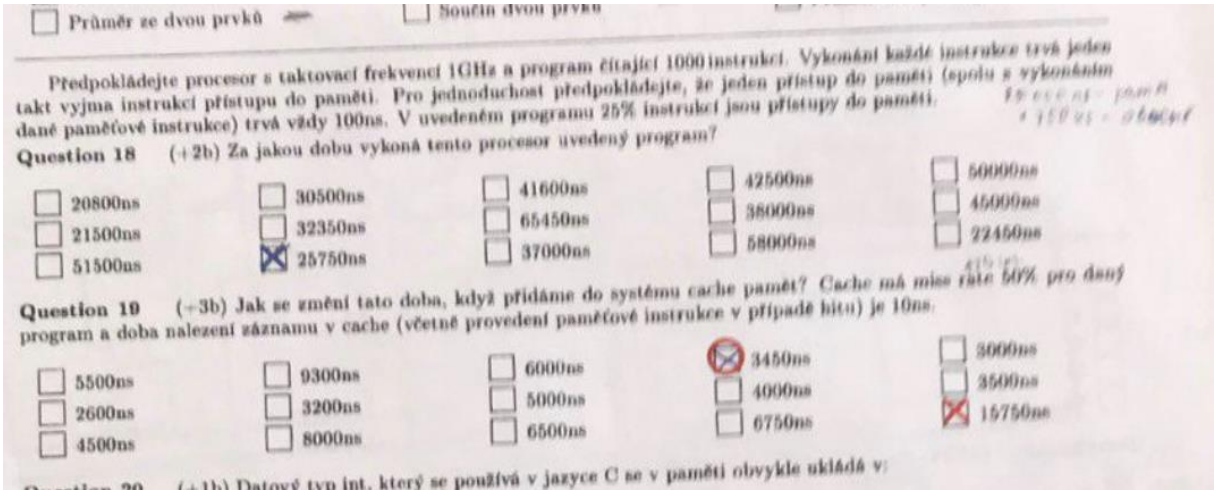
Máme dvě 8-bitová čísla - A a B
Čísla sečteme: 101111
\+ 11100100
100010011

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 ... pardon, nerozumím zápisu té logické funkce, napište kdo ví pls:D

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.



Průměr ze dvou prvků — Součin dvou prvků

Předpokládejte procesor s taktovací frekvencí 1GHz a program čítající 1000 instrukcí. Vykonání každé instrukce trvá jeden takt vyjma instrukcí přístupu do paměti. Pro jednoduchost předpokládejte, že jeden přístup do paměti (spolu s vykonáním dané paměťové instrukce) trvá vždy 100ns. V uvedeném programu 25% instrukcí jsou přístupy do paměti.

Question 18 (+2b) Za jakou dobu vykoná tento procesor uvedený program?

☐ 20800ns	☐ 30500ns	☐ 41600ns	☐ 42500ns	☐ 50000ns
☐ 21500ns	☐ 32350ns	☐ 65450ns	☐ 38000ns	☐ 45000ns
☐ 51500ns	☒ 25750ns	☐ 37000ns	☐ 58000ns	☐ 22450ns

Question 19 (+3b) Jak se změní tato doba, když přidáme do systému cache paměť? Cache má miss rate 50% pro daný program a doba nalezení záznamu v cache (včetně provedení paměťové instrukce v případě hitu) je 10ns.

☐ 5500ns	☐ 9300ns	☐ 6000ns	☒ 3450ns	☐ 3000ns
☐ 2600ns	☐ 3200ns	☐ 5000ns	☐ 4000ns	☐ 3500ns
☐ 4500ns	☐ 8000ns	☐ 6500ns	☐ 6750ns	☒ 15750ns

(+1b) Datový typ int, který se používá v jazyce C se v paměti obvykle ukládá v:

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

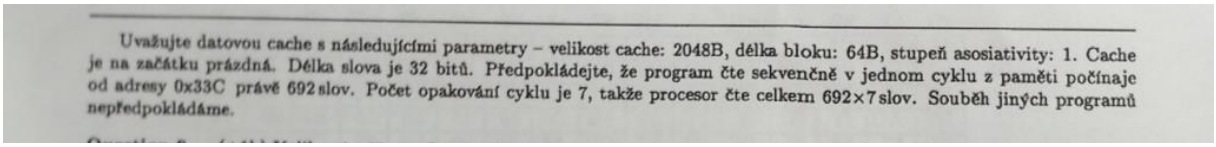
1) Za jakou dobu vykoná tento procesor uvedený program?

$$0.25 \times 1000 = 250$$
$$\tau = 250 \times 100 + 750 \times 1 = 25750\,ns$$

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.

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

CACHE



Uvažujte datovou cache s následujícími parametry – velikost cache: 2048B, délka bloku: 64B, stupeň asosiativity: 1. Cache je na začátku prázdná. Délka 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

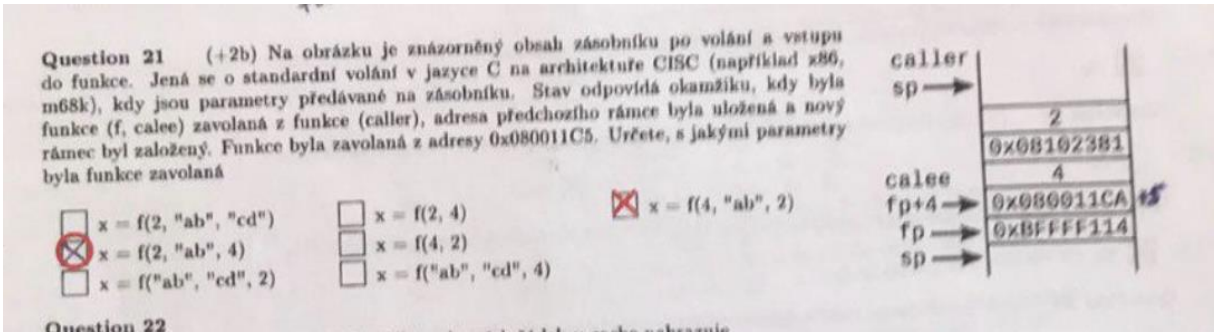
1) Kolik má cache setů?
Spočítám podle vzorce
počet setů S = $\frac{velikost\ cache}{délka\ bloku \times stupeň\ asociativity} = \frac{2048}{64 \times 1} = 32\ setů$

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:
12 % 32 = 12
Takže data se namapují do setu číslo 12.

3) Počet missů / hitů cache
Bohužel pro vícecestnou cache nevím, jak na to, zvládám jen jednocestnou

ARGUMENTY FUNKCE



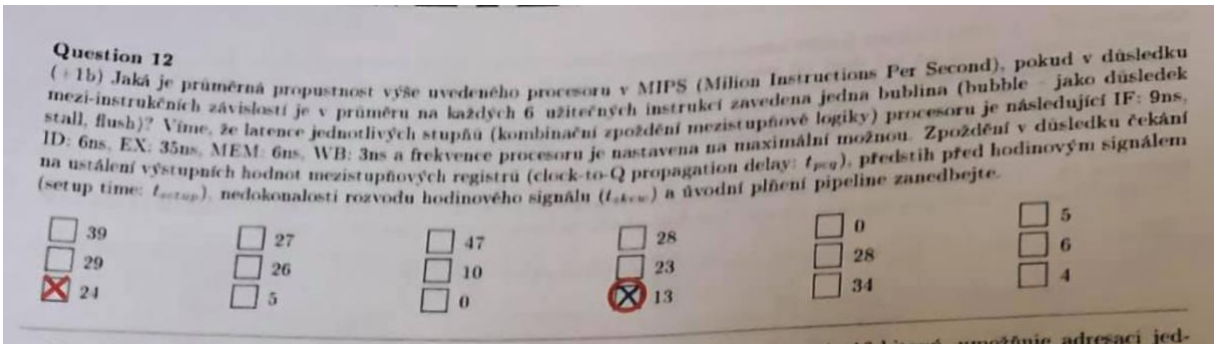
Question 21 (+2b) Na obrázku je znázorněný obsah zásobníku po volání a vstupu do funkce. Jená se o standardní volání v jazyce C na architektuře CISC (například x86, m68k), kdy jsou parametry předávané na zásobníku. Stav odpovídá okamžiku, kdy byla funkce (f, calee) zavolaná z funkce (caller), adresa předchozího rámce byla uložená a nový rámec byl založený. Funkce byla zavolaná z adresy 0x080011C5. Určete, s jakými parametry byla funkce zavolaná

- ☐ x = f(2, "ab", "cd")
- ☒ x = f(2, "ab", 4)
- ☐ x = f("ab", "cd", 2)
- ☐ x = f(2, 4)
- ☐ x = f(4, 2)
- ☐ x = f("ab", "cd", 4)
- ☒ x = f(4, "ab", 2)

Question 22

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



Question 12
(+1b) Jaká je průměrná propustnost výše uvedeného procesoru v MIPS (Milion Instructions Per Second), pokud v důsledku mezi-instrukčních závislostí je v průměru na každých 6 užitečných instrukcí zavedena jedna bublina (bubble - jako důsledek stall, flush)? Víme, že latence jednotlivých stupňů (kombinační zpoždění mezistupňové logiky) procesoru je následující IF: 9ns, ID: 6ns, EX: 35ns, MEM: 6ns, WB: 3ns a frekvence procesoru je nastavena na maximální možnou. Zpoždění v důsledku čekání na ustálení výstupních hodnot mezistupňových registrů (clock-to-Q propagation delay: t_{pcq}), předstih před hodinovým signálem (setup time: t_{setup}), nedokonalosti rozvodu hodinového signálu (t_{skew}) a úvodní plnění pipeline zanedbejte.

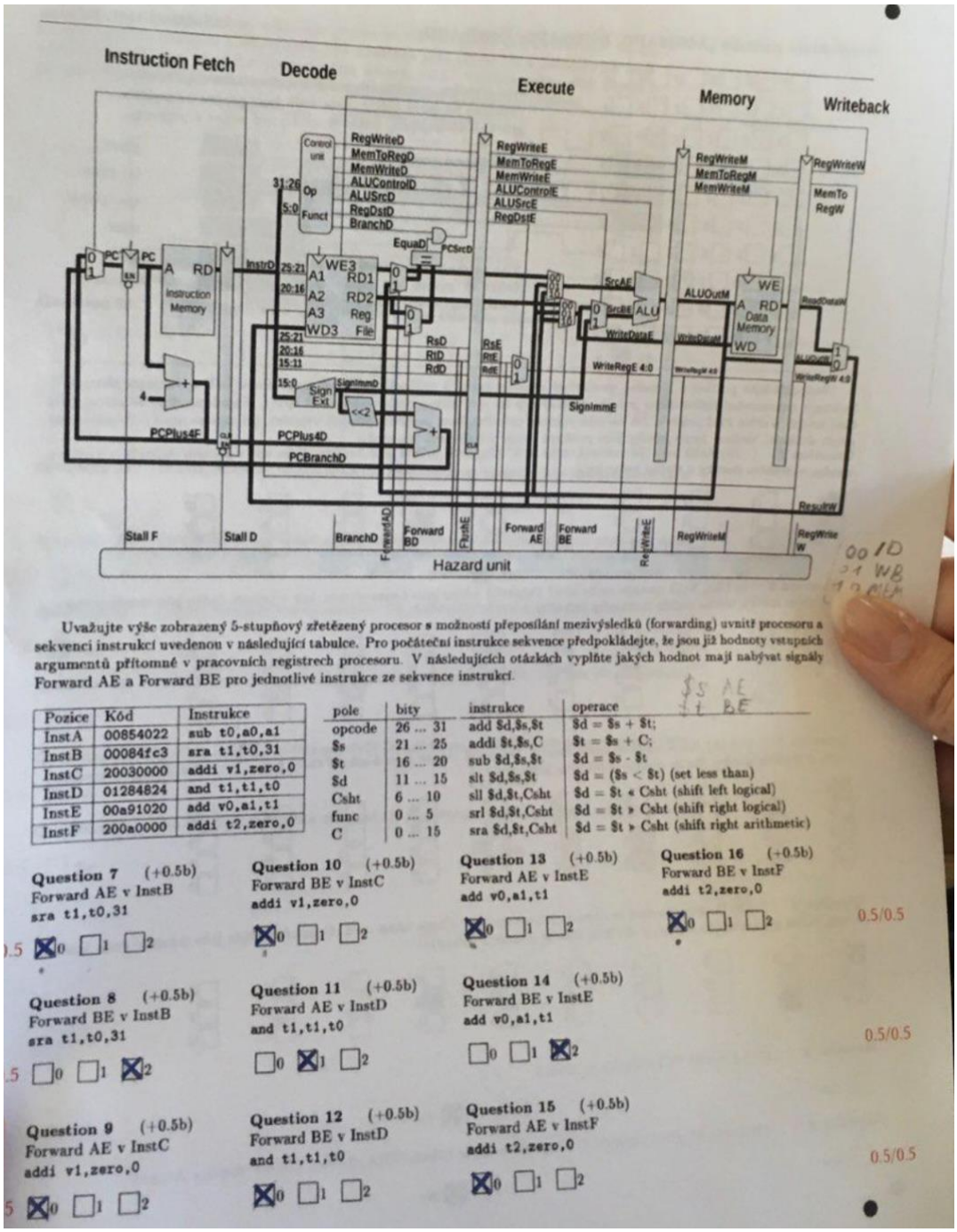
- ☐ 39
- ☐ 29
- ☒ 24
- ☐ 27
- ☐ 26
- ☐ 5
- ☐ 47
- ☐ 10
- ☐ 0
- ☐ 28
- ☐ 23
- ☒ 13
- ☐ 0
- ☐ 28
- ☐ 34
- ☐ 5
- ☐ 6
- ☐ 4

Zjistím poměr užitečných instrukcí / bublina ze zadání
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

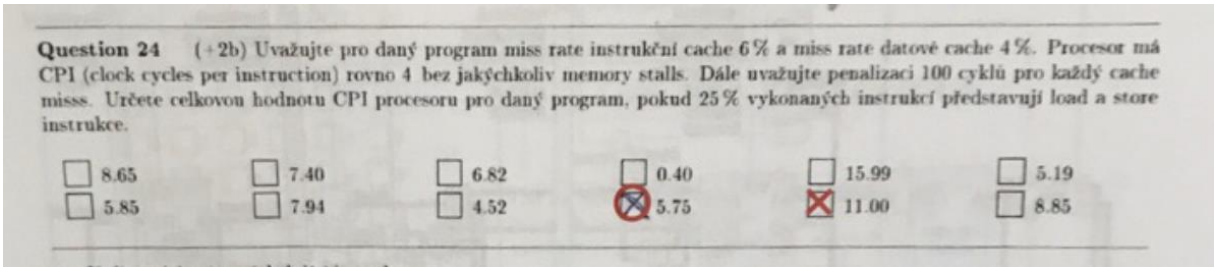
1 s / 35 ns = 28,6 MHz
28,6 MHz x 5/6 = 23.8 = 24

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



Question 24 (+2b) Uvažujte pro daný program miss rate instrukční cache 6 % a miss rate datové cache 4 %. Procesor má CPI (clock cycles per instruction) rovno 4 bez jakýchkoliv memory stalls. Dále uvažujte penalizaci 100 cyklů pro každý cache misss. Určete celkovou hodnotu CPI procesoru pro daný program, pokud 25 % vykonaných instrukcí představují load a store instrukce.

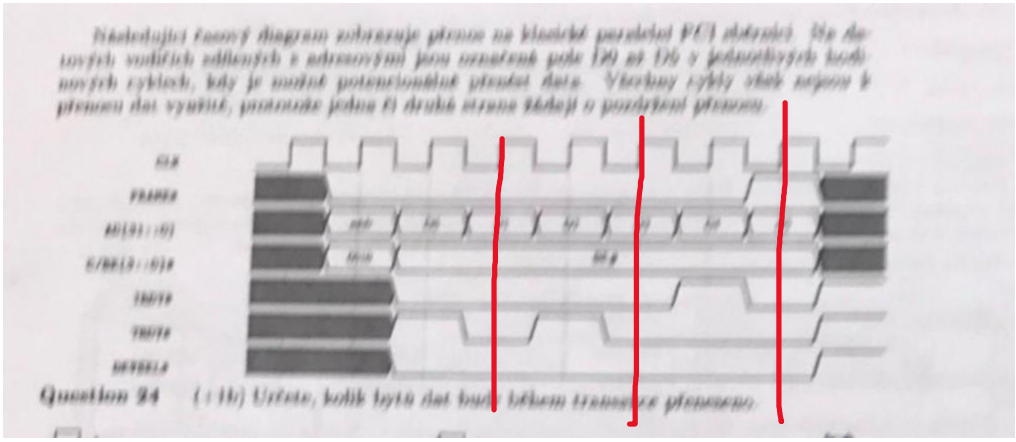
☐ 8.65	☐ 7.40	☐ 6.82	☐ 0.40	☐ 15.99	☐ 5.19
☐ 5.85	☐ 7.94	☐ 4.52	☒ 5.75	☒ 11.00	☐ 8.85

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 x 1 x 100) + (0.04 x 0.25 x 100) = 4 + 6 + 1 = 11

DATOVÝ PŘENOS NA SBĚRNICI PCI



Question 24 (+1b) Určete, kolik bytů dat bude během transakce přeneseno.

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 3 byty - 32 bitů
sorry for bad image _(ツ)_/¯

3 x 4 = 12 -bude přeneseno 12 Bytů

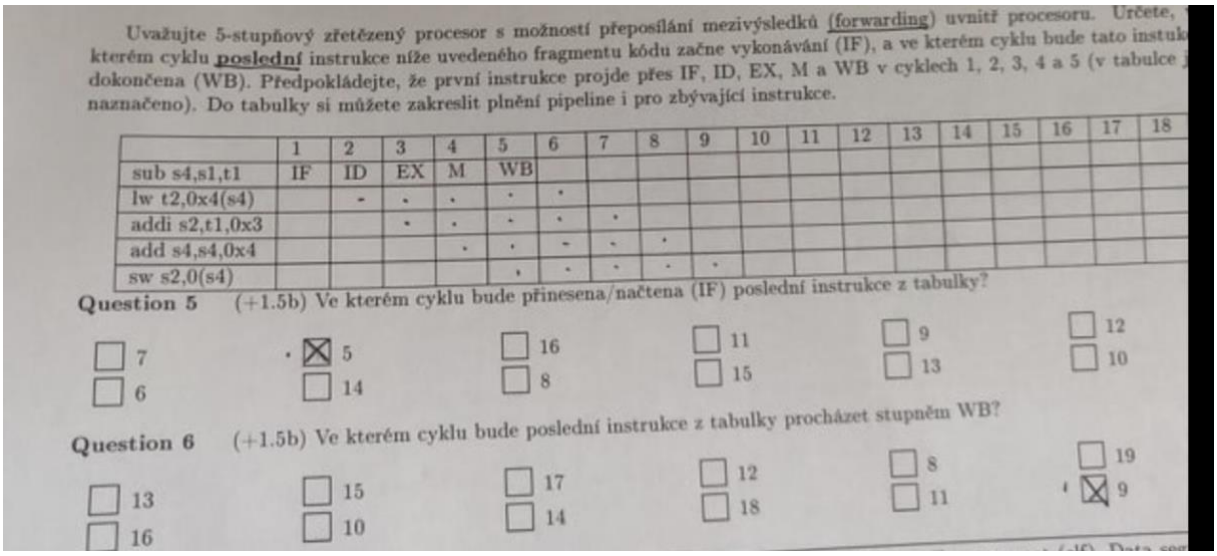
2) Hodnoty vodičů označené D0 (D1, D2, ... Dn) představují okamžik, kdy jsou přenášená data

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



Uvažujte 5-stupňový zřetězený procesor s možností přeposílání mezivýsledků (forwarding) uvnitř procesoru. Určete, v kterém cyklu **poslední** instrukce níže uvedeného fragmentu kódu začne vykonávání (IF), a ve kterém cyklu bude tato instrukce dokončena (WB). Předpokládejte, že první instrukce projde přes IF, ID, EX, M a WB v cyklech 1, 2, 3, 4 a 5 (v tabulce je to naznačeno). Do tabulky si můžete zakreslit plnění pipeline i pro zbývající instrukce.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
sub s4,s1,t1	IF	ID	EX	M	WB													
lw t2,0x4(s4)		IF	ID	EX	M	WB												
addi s2,t1,0x3			IF	ID	EX	M	WB											
add s4,s4,0x4				IF	ID	EX	M	WB										
sw s2,0(s4)					IF	ID	EX	M	WB									

Question 5 (+1.5b) Ve kterém cyklu bude přinesena/načtena (IF) poslední instrukce z tabulky?

☐ 7 ☒ 5 ☐ 16 ☐ 11 ☐ 9 ☐ 12
☐ 6 ☐ 14 ☐ 8 ☐ 15 ☐ 13 ☐ 10

Question 6 (+1.5b) Ve kterém cyklu bude poslední instrukce z tabulky procházet stupněm WB?

☐ 13 ☐ 15 ☐ 17 ☐ 12 ☐ 8 ☐ 19
☐ 16 ☐ 10 ☐ 14 ☐ 18 ☐ 11 ☒ 9

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

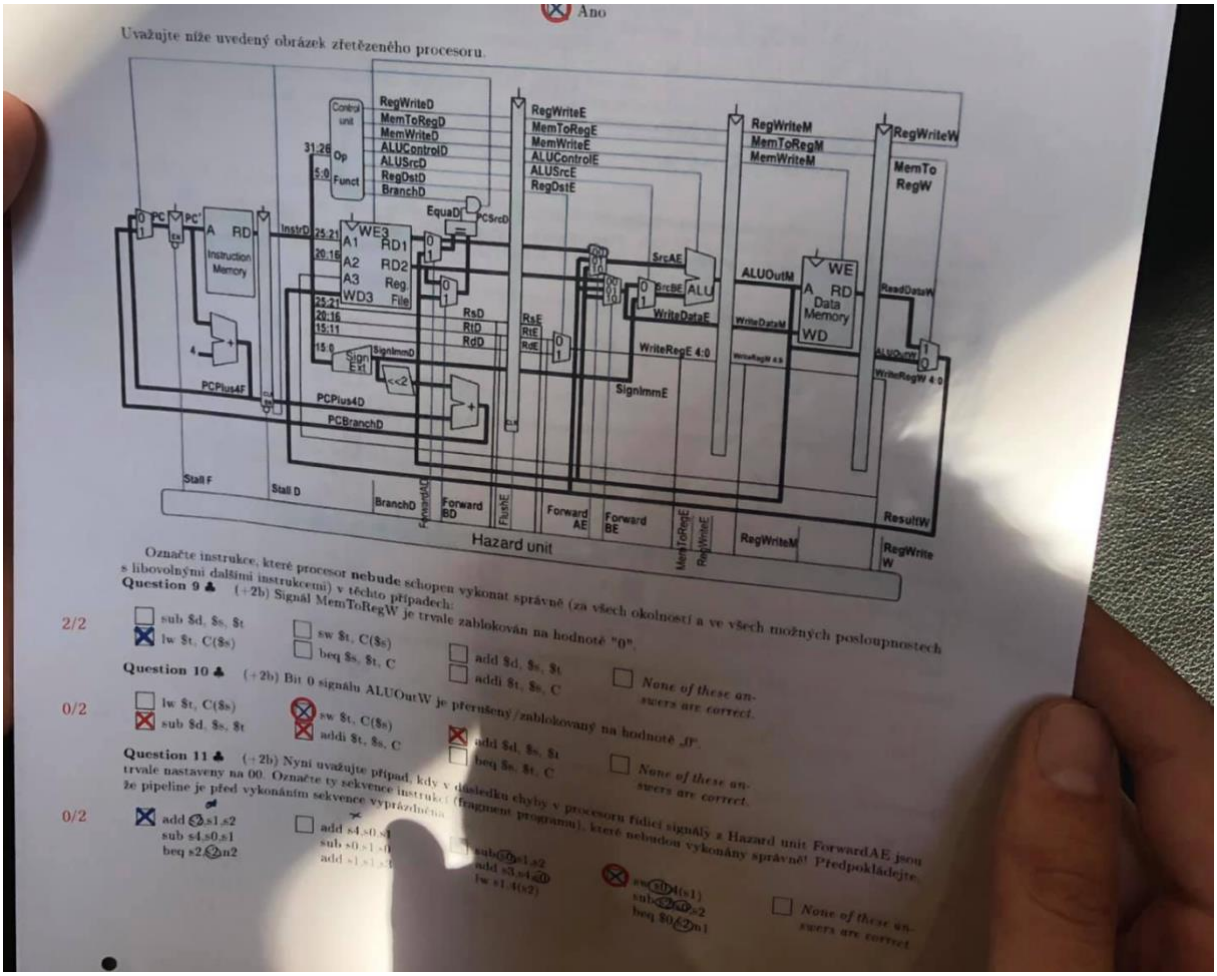
s forwardingem - záleží na typu instrukcí

- -mezi aritmetickými instrukcemi forwarding vyřeší všechno, nemusím dávat žádné stally
- -mezi aritmetickou a paměťovou (lw, sw) musím dát jeden stall, pokud je tam závislost
- -u branch instrukci je to slozitejsi, nejsem si jista.

btw. není to opravené, takže zakřížkované odpovědi nejsou správně

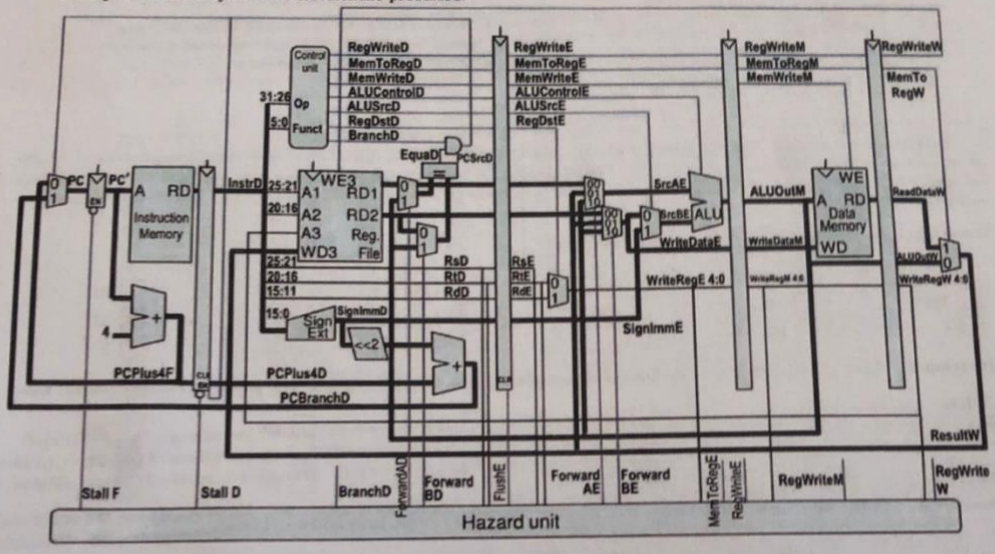
Kdo ví jak na tuto úlohu, podělte se.

BETTER LEARN THIS



-

Uvažujte níže uvedený obrázek zřetězeného procesoru.



Označte instrukce, které procesor **nebude** schopen vykonat správně (za všech okolností a ve všech možných posloupnostech s libovolnými dalšími instrukcemi) v těchto případech:

Question 9 ♣ (+2b) Jednotka SignExt nastaví horních 16 bitů na výstupu vždy na 0.

- [x] beq $s, $t, C
- [x] lw $t, C($s)
- [x] sw $t, C($s)
- [x] add $d, $s, $t
- [x] sub $d, $s, $t
- [x] addi $t, $s, C
- [ ] *None of these answers are correct.*

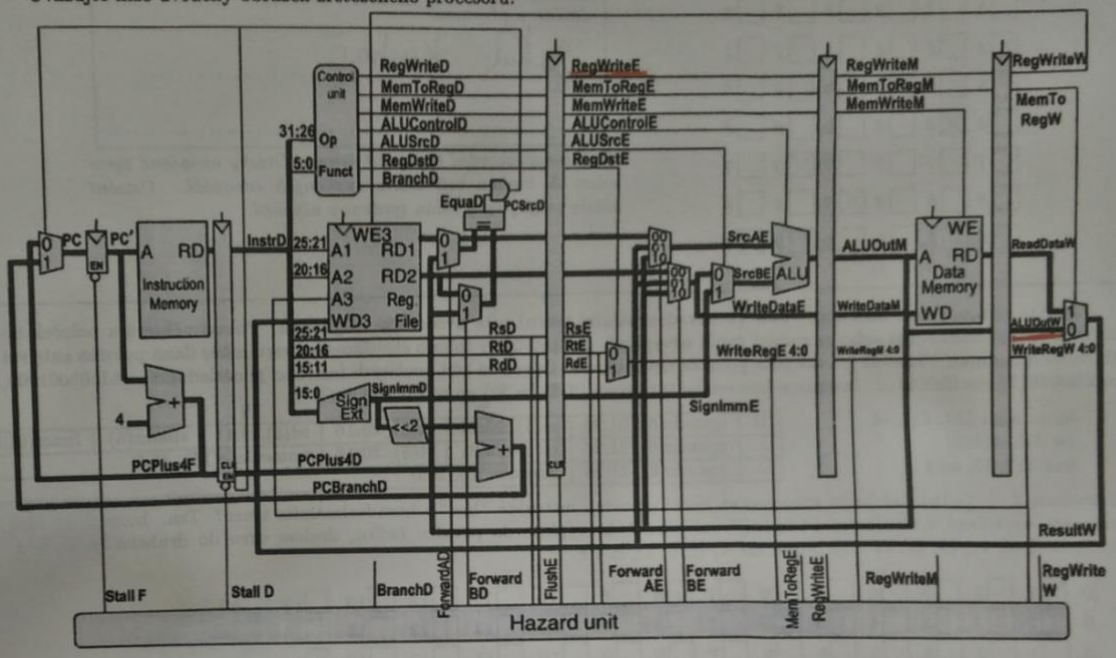
Question 10 ♣ (+2b) Nejnižší bit signálu WriteDataE je přerušený/zablokovaný na hodnotě „0".

- [ ] addi $t, $s, C
- [x] lw $t, C($s)
- [ ] sub $d, $s, $t
- [x] sw $t, C($s)
- [ ] beq $s, $t, C
- [ ] add $d, $s, $t
- [ ] *None of these answers are correct.*

Question 11 ♣ (+2b) Nyní uvažujte případ, kdy v důsledku chyby v procesoru řídicí signály z Hazard unit ForwardAE jsou trvale nastaveny na 00. Označte ty sekvence instrukcí (fragment programu), které nebudou vykonány správně! Předpokládejte, že pipeline je před vykonáním sekvence vyprázdněna.

- [ ] addi s0,$0,1
sub s2,s1,s2
add s1,s1,s3
- [x] add s0,s1,s1
sub s2,s0,s2
beq $0,s2,n1
- [ ] sub s0,s1,s2
add s3,s4,s0
addi s1,s2,2
- [x] add s0,s1,s2
sub s4,s0,s1
beq s2,s2,n2
- [ ] *None of these answers are correct.*

Uvažujte níže uvedený obrázek zřetězeného procesoru.



Označte instrukce, které procesor **nebude** schopen vykonat správně (za všech okolností a ve všech možných posloupnostech s libovolnými dalšími instrukcemi) v těchto případech:

Question 6 ♣ (+2b) Bit 0 signálu ALUOutW je přerušený/zablokovaný na hodnotě „0".

- [ ] beq $s, $t, C
- [x] sub $d, $s, $t
- [x] addi $t, $s, C
- [ ] lw $t, C($s)
- [ ] sw $t, C($s)
- [x] add $d, $s, $t
- [ ] *None of these answers are correct.*

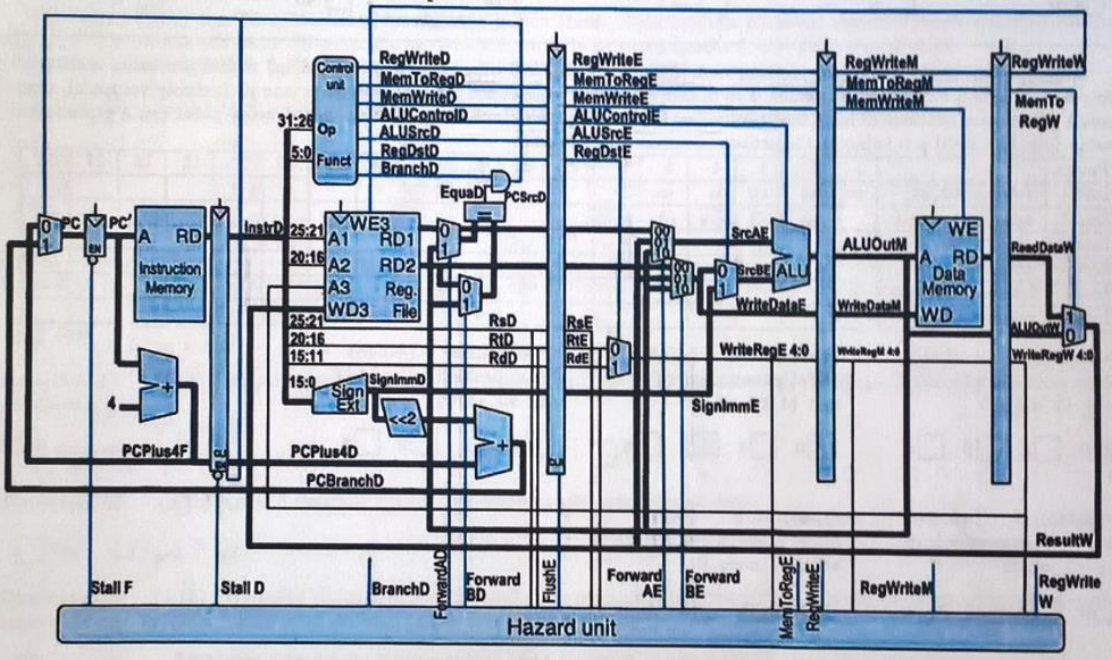
Question 7 ♣ (+2b) Signál RegWriteE je přerušený/zablokovaný na hodnotě „0".

- [x] lw $t, C($s)
- [x] sub $d, $s, $t
- [x] beq $s, $t, C
- [x] sw $t, C($s)
- [x] add $d, $s, $t
- [x] addi $t, $s, C
- [ ] *None of these answers are correct.*

Question 8 ♣ (+2b) Nyní uvažujte případ, kdy v důsledku chyby v procesoru řídicí signály z Hazard unit ForwardAE jsou trvale nastaveny na 00. Označte ty sekvence instrukcí (fragment programu), které nebudou vykonány správně! Předpokládejte, že pipeline je před vykonáním sekvence vyprázdněna.

- [x] add s2,s1,s2 / sub s4,s0,s1 / beq s2,s2,n2
- [x] sub s0,s1,s2 / add s3,s4,s0 / lw s1,4(s2)
- [ ] add s4,s0,s1 / sub s0,s1,s0 / add s1,s1,s3
- [x] sw s0,4(s1) / sub s2,s0,s2 / beq $0,s2,n1
- [ ] *None of these answers are correct.*

Uvažujte níže uvedený obrázek zřetězeného procesoru.



Označte instrukce, které procesor **nebude** schopen vykonat správně (za všech okolností a ve všech možných posloupnostech s libovolnými dalšími instrukcemi) v těchto případech:

Question 1 ♣ (+2b) Signál ForwardBD je přerušený/zablokovaný na hodnotě „0".

- ☐ lw $t, C($s)
- ☐ add $d, $s, $t
- ☐ addi $t, $s, C
- ☐ sub $d, $s, $t
- ☒ beq $s, $t, C
- ☐ sw $t, C($s)
- ☒ *None of these answers are correct.*

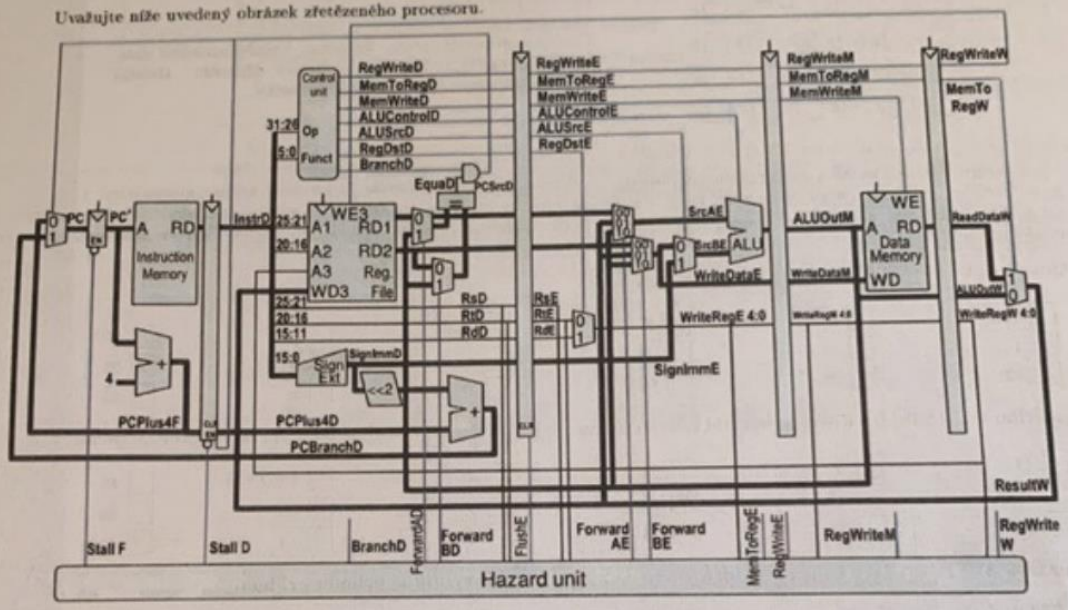
Question 2 ♣ (+2b) Signál MemToRegW je trvale zablokován na hodnotě "0".

- ☒ sw $t, C($s)
- ☒ addi $t, $s, C
- ☒ lw $t, C($s)
- ☒ add $d, $s, $t
- ☒ beq $s, $t, C
- ☒ sub $d, $s, $t
- ☐ *None of these answers are correct.*

Question 3 ♣ (+2.5b) Nyní uvažujte případ, kdy v důsledku chyby v procesoru řídicí signály z Hazard unit ForwardAE mají nejvyšší bit trvale nastaven na 0. Jinými slovy, pokud ForwardAE má být 00, 01, 10, pak bude 00, 01, 00. Označte ty sekvence instrukcí (fragment programu), které nebudou vykonány správně! Předpokládejte, že pipeline je před vykonáním sekvence vyprázdněna.

- ☒ beq $0,s2,n1
add s0,s1,s1
sub s2,s0,s2
- ☐ sub s2,s1,s2
addi s0,$0,1
add s1,s1,s3
- ☒ sub s0,s1,s2
add s3,s4,s0
addi s1,s0,2
- ☐ sub s2,s3,s2
sub s4,s0,s1
add s2,s2,s2
- ☐ *None of these answers are correct.*

Uvažujte níže uvedený obrázek zřetězeného procesoru.



Označte instrukce, které procesor **nebude** schopen vykonat správně (za všech okolností a ve všech možných posloupnostech s libovolnými dalšími instrukcemi) v těchto případech:

Question 9 ♣ (+2b) Nejnižší bit signálu WriteDataE je přerušený/zablokovaný na hodnotě „0".

2/2

- ☐ beq $s, $t, C
- ☐ sub $d, $s, $t
- ☐ addi $t, $s, C
- ☐ *None of these answers are correct.*
- ☒ sw $t, C($s)
- ☐ add $d, $s, $t
- ☐ lw $t, C($s)

Question 10 ♣ (+2b) Signál ALUSrcE je přerušený/zablokovaný na hodnotě „0".

0/2

- ☒ sw $t, C($s)
- ☐ beq $s, $t, C
- ☒ lw $t, C($s)
- ☐ *None of these answers are correct.*
- ☒ addi $t, $s, C
- ☒ sub $d, $s, $t
- ☒ add $d, $s, $t

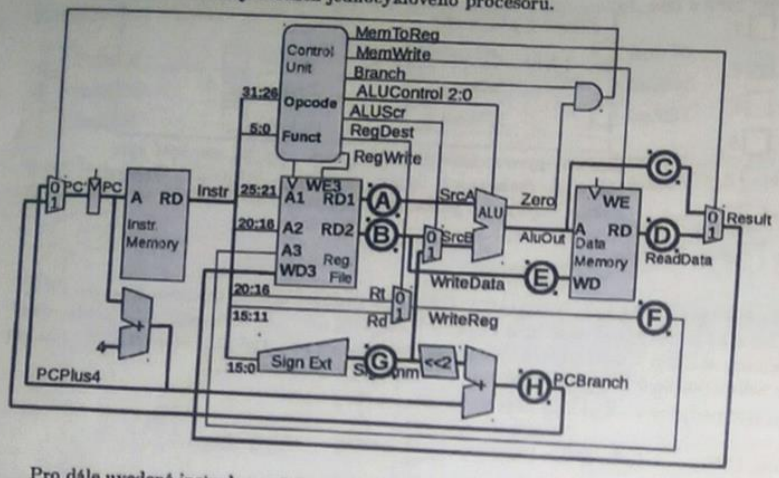
Question 11 ♣ (+2b) Nyní uvažujte případ, kdy v důsledku chyby v procesoru řídicí signály z Hazard unit ForwardAE jsou trvale nastaveny na 00. Označte ty sekvence instrukcí (fragment programu), které nebudou vykonány správně! Předpokládejte, že pipeline je před vykonáním sekvence vyprázdněna.

0/2

- ☐ sub s0,s1,s2 / add s3,s4,s0 / lw s1,4(s2) — ForwardBE
- ☒ sw s0,4(s1) / sub s2,s0,s2 / beq $0,s2,n1 — ForwardAE 10
- ☐ add s4,s0,s1 / sub s0,s1,s0 / add s1,s1,s3 — Nema forw
- ☒ add s2,s1,s2 / sub s4,s0,s1 / beq s2,s2,n2 — Forward AE 01
- ☐ *None of these answers are correct.*

☐ 0.031 ☒ 0.016 ☐ 0.000

Uvažujte níže uvedený obrázek jednocyklového procesoru.



Pro dále uvedené instrukce určete všechny spoje/signály, které daná instrukce bude aktivovat. Počítají se právě ty signály a datové cesty, pro které by se případná změna/rozpojení projevilo na výsledném stavu CPU po provedení dané instrukce alespoň s některou kombinací parametrů.

Question 10 ♣ (+1b) Které spoje budou aktivní pro instrukci `lw $t,C($s)`.

1/1 ☒ A ☐ B ☐ C ☒ D ☐ E ☒ F ☒ G ☐ H

Question 11 ♣ (+1b) Které spoje budou aktivní pro instrukci `add $d,$s,$t`.

/1 ☒ A ☒ B ☒ C ☐ D ☐ E ☒ F ☐ G ☐ H

Question 12 ♣ (+1b) Které spoje budou aktivní pro instrukci `sw $t,C($s)`.

)/1 ☒ A ☒ B ☒ C ☐ D ☒ E ☐ F ☒ G ☐ H

Question 13 ♣ (+1b) Které spoje budou aktivní pro instrukci `slt $d,$s,$t`.

/1 ☒ A ☒ B ☒ C ☐ D ☐ E ☒ F ☐ G ☐ H

Uvažuje následující sekvenci instrukcí