Předpokládejte branch delay slot 1 cyklus a uvažujte níže uvedený program. Délka instrukce je fixní a je 32 bitů. Dále předpokládejte 1-cestné asociativní (přímo mapovanou) instrukční cache o velikosti 64B, přičemž délka bloku je 8B. Cache je na začátku prázdná a strategie nahrazování je LRU.

adresa	instrukce
0x0000 000C	addi s1, $0, 0
0x0000 0010	addi t0, $0, 5
0x0000 0014	L: addi t0, t0, -1
0x0000 0018	bne t0, $0, L
0x0000 001C	addi s1, $0, 7

Question 1 (+2b) Kolik má výše uvedená cache setů?

9	17	0	19	12
15	10	18	16	14
20	11	8 •	21	13

Question 2 (+3b) Uveďte číslo setu, do kterého se namapuje instrukce **bne** výše uvedeného programu. Pozor, první set má 0 (index) nula

0/3

1	10	8	13	4
3	15	5	12	9
2 •	7	14	11	6

Question 3 (+2b) Pokud vykonáme výše uvedený program (Program Counter definitivně opustí cyklus), jaký bude celkový počet **hitů** v cache? Nezapomeňte na delay slot 1 cyklus skokové instrukce.

0/2

16	0	15	12	11
9 •	39	19	10	29
13	14	1	21	24

Question 4 (+2b) Pokud vykonáme výše uvedený program (Program Counter definitivně opustí cyklus), jaký bude celkový počet **misů** v cache? Nezapomeňte na delay slot 1 cyklus skokové instrukce.

0/2

18	1	3	4	5 •
14	10	0	7	2
1	8	33	13	2

Uvažujte typické vlastnosti pro jednotlivé generace a typy architektur CPU. V případě 8-bit CPU uvažujte koncepci z let 71 až 75, například Intel 8080. 16-bit CISC architektury jsou například původní Intel 8086 a koncepcí i m68k. Jako představitelé RISC architektur byly vybrané architektury MIPS a SPARC, které byly v této fázi vývoje průkopníky. V následujících otázkách

je použita zkratka ALop pro základní aritmetickologické operace (sčítání, odčítání, logický součin atd.).

Question 5 ♣ (+0.5b) Veškeré instrukce v základním režimu mají jednu pevnou délku.

0/0.5

- [ ] historická 8-bit CPU
- [ ] 16-bit CISC
- [ ] MIPS •
- [ ] SPARC

Question 6 ♣ (+0.5b) Přenosy dat mezi pamětí a registry obstarává omezená skupina instrukcí (load/store).

0/0.5

- [ ] historická 8-bit CPU
- [ ] 16-bit CISC
- [ ] MIPS •
- [ ] SPARC

Question 7 ♣ (+0.5b) ALop je možné provozovat pouze mezi operandy v registrech.

0/0.5

- [ ] historická 8-bit CPU
- [ ] 16-bit CISC
- [ ] MIPS •
- [ ] SPARC

Question 8 ♣ (+0.5b) Pro uložení návratové adresy se používá specializovaný registr.

0/0.5

- [ ] historická 8-bit CPU
- [ ] 16-bit CISC
- [ ] MIPS
- [ ] SPARC

Question 9 (±1b) RISC procesory používají tzv. Load/Store model, v němž nelze v jedné instrukci kombinovat paměťové a aritmetické (ALU) operace – což neplatí pro procesory CISC.

1/1

- [ ] Ne
- [ ] Ano •

Question 10 (±1b) Program/OS se během DMA čtení nebo zápisu věnuje jiným operacím. Pouze naplánuje parametry přenosu a nastaví adresy do DMA řadiče – ten po ukončení operace vyvolá přerušení.

1/1

- [ ] Ano
- [ ] Ne

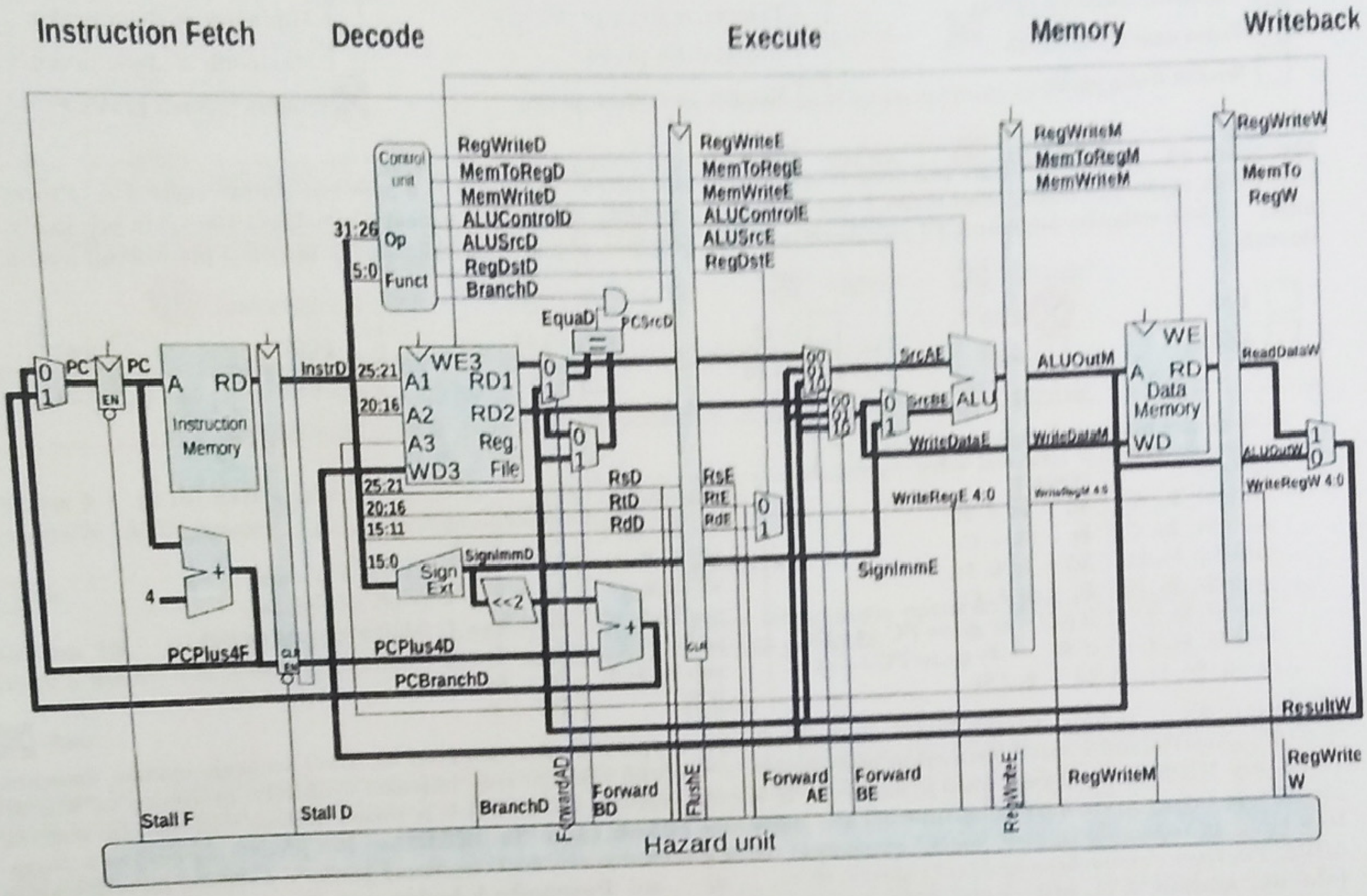
Question 11 (±1b) 64bitové procesory používají 64-bitový formát virtuální adresy, ale jejich virtuální stránkování využívá menší počet bitu (zpravidla 48). Důvodem je zmenšení počtu úrovní stránkovacích tabulek.

0/1

- [ ] Ano
- [ ] Ne

Question 12 (±1b) 32bitové integer čísla se znaménkem ve dvojkovém doplňku ukládají menší počet číselných hodnot než čísla bez znaménka.

- [ ] Ne
- [ ] Ano



Uvažujte výše zobrazený 5-stupňový zřetězený procesor s možností přeposílání mezivýsledků (forwarding) uvnitř procesoru a sekvenci instrukcí uvedenou v následující tabulce. Pro počáteční instrukce sekvence předpokládejte, že jsou již hodnoty vstupních argumentů přítomné v pracovních registrech procesoru. V následujících otázkách vyplňte jakých hodnot mají nabývat signály Forward AE a Forward BE pro jednotlivé instrukce ze sekvence instrukcí.

Pozice	Kód	Instrukce
InstA	20ca0000	addi t2,a2,0
InstB	20a90000	addi t1,a1,0
InstC	20eb0000	addi t3,a3,0
InstD	00896020	add t4,a0,t1
InstE	014b6820	add t5,t2,t3
InstF	018d1020	add v0,t4,t5

pole	bity	instrukce	operace
opcode	26 ... 31	add $d,$s,$t	$d = $s + $t;
$s	21 ... 25	addi $t,$s,C	$t = $s + C;
$t	16 ... 20	sub $d,$s,$t	$d = $s - $t
$d	11 ... 15	slt $d,$s,$t	$d = ($s < $t) (set less than)
Csht	6 ... 10	sll $d,$t,Csht	$d = $t « Csht (shift left logical)
func	0 ... 5	srl $d,$t,Csht	$d = $t » Csht (shift right logical)
C	0 ... 15	sra $d,$t,Csht	$d = $t » Csht (shift right arithmetic)

Question 13 (+0.5b)
Forward AE v InstB
`addi t1,a1,0`

0.5/0.5 0 1 2

Question 14 (+0.5b)
Forward BE v InstB
`addi t1,a1,0`

0.5/0.5 0 1 2

Question 15 (+0.5b)
Forward AE v InstC
`addi t3,a3,0`

0/0.5 0 1 2

Question 16 (+0.5b)
Forward BE v InstC
`addi t3,a3,0`

0 1 2

Question 17 (+0.5b)
Forward AE v InstD
`add t4,a0,t1`

0 1 2

Question 18 (+0.5b)
Forward BE v InstD
`add t4,a0,t1`

0 1 2

Question 19 (+0.5b)
Forward AE v InstE
`add t5,t2,t3`

0 1 2

Question 20 (+0.5b)
Forward BE v InstE
`add t5,t2,t3`

0 1 2

Question 21 (+0.5b)
Forward AE v InstF
`add v0,t4,t5`

0 1 2

Question 22 (+0.5b)
Forward BE v InstF
`add v0,t4,t5`

0 1 2

0.0/0.5

0/0

0/0

Question 23 (+2b) Jakou funkci/výpočet realizuje fragment kódu analyzovaný v předchozích otázkách?

2/2

Nejmenší společný dělitel	Průměr ze dvou prvků	Minimum ze dvou prvků
Porovnání dvou prvků	Rozdíl dvou prvků	Maximum ze dvou prvků
Součin dvou prvků	Průměr ze čtyřech prvků	Součet čtyřech prvků •

Question 24 (+2b) Uvažujte pro daný program miss rate instrukční cache 7 % a miss rate datové cache 7 %. 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 30 % vykonaných instrukcí představují load a store instrukce.

0/2

4.53	5.96 •	5.12	15.45	24.07	9.67
5.63	5.33	18.75	8.28	13.10	4.54

K dispozici máte následující instrukce.

instrukce	operace	instrukce	operace
add $d, $s, $t	$d = $s + $t;	ori $t, $s, C	$t = $s | C;
addi $t, $s, C	$t = $s + C;	slt $d,$s,$t	$d = ($s < $t) (set less than)
and $d, $s, $t	$d = $s & $t;	sll $d,$s,C	$d = $s « C (shift left logical)
andi $t, $s, C	$t = $s & C;	sra $d,$s,C	$d = $s » C (shift right aritmetic)
bne $s, $t, C	if $s != $t go to PC+4+4*C	srl $d,$s,C	$d = $s » C (shift right logical)
beq $s, $t, C	if $s == $t go to PC+4+4*C	sub $d,$s,$t	$d = $s - $t
or $d, $s, $t	$d = $s | $t;	jr $s	PC = $s

Question 25 (+6b) Uvažujte funkci dle funkčního prototypu v C: *float pow2(int x)*, která realizuje operaci umocnění $y = 2^x$. Napište kód v assembleru (jako podprogram) s využitím základní sady instrukcí výukového procesoru na bázi architektury MIPS. Vstupní **celé číslo** je uloženo v 32-bitovém registru, výstupní číslo je rovněž uloženo v 32-bitovém registru, ale interně reprezentuje **normalizované číslo v plovoucí řádové čárce dle IEEE754**. Pro předání parametru a návratové hodnoty v assembleru bude použitá standardní volací konvence používaná kompilátory jazyka C a dalšími na architektuře MIPS. Používejte symbolické označení registrů, t0, t1, ..., s0, ... atd. **Poznámka k hodnocení: 2b za vyjádření myšlenky (slovně, kódem v C, atd. nebo přímo v assembleru), 3b za rychlost řešení, 3b za program v assembleru**

[redacted] 1/6