Solutions Exam E2017

Problem 1

Question 1.1

32 bits.

Question 1.2

To forward computed values to data dependent instructions before the values are written back into the register file.

Question 1.3

a) **Register fields are always in the same position in the instruction to simplify pipelining.**

b) RISC-V can add two numbers that are in memory in a single instruction.

c) A single-cycle datapath is the best performing implementation of RISC-V.

Question 1.4

Because code and data have temporal and spatial locality.

Question 1.5

True: An instruction set simulator of RISC-V can boot Linux.

... continued on the next page

Problem 2

Question 2.1

P1:

		1	2	3	4	5	6	7	8	9
main:										
	addi a2, x0, 5	X								
	lw a1, 0(a0)		X							
	addi a1, a1, 5			X						
	sw a1, 0(a0)				X					
	add a1, x0, a2					X				
	beq a1, x0, branch1						X			
	beq a1, a2, branch2							X		
branch1:										
	add a1, a1, a1									
	add a2, a2, a2									
branch2:										
	addi a1, a1, 6								X	
	addi a2, a2, 7									X

P2:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
main:																					
	addi a2, x0, 5	F	D	E	M	W															
	lw a1, 0(a0)		F	D	E	M	W														
	addi a1, a1, 5			F	-	-	D	E	M	W											
	sw a1, 0(a0)				-	-	F	-	-	D	E	M	W								
	add a1, x0, a2							-	-	F	D	E	M	W							
	beq a1, x0, branch1										F	-	-	D	E	M	W				
	beq a1, a2, branch2											-	-	F	D	E	M	W			
branch1:																					
	add a1, a1, a1														F						
	add a2, a2, a2																				
branch2:																					
	addi a1, a1, 6															F	D	E	M	W	
	addi a2, a2, 7																F	D	E	M	W

... continued on the next page

P3:

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
main:																		
	addi a2, x0, 5	F	D	E	M	W												
	lw a1, 0(a0)		F	D	E	M	W											
	addi a1, a1, 5			F	-	D	E	M	W									
	sw a1, 0(a0)				-	F	-	D	E	M	W							
	add a1, x0, a2						-	F	D	E	M	W						
	beq a1, x0, branch1								F	-	D	E	M	W				
	beq a1, a2, branch2									-	F	D	E	M	W			
branch1:																		
	add a1, a1, a1											F						
	add a2, a2, a2																	
branch2:																		
	addi a1, a1, 6												F	D	E	M	W	
	addi a2, a2, 7													F	D	E	M	W

Question 2.2

$$T_{\mathbf{P1}} = 9 * 10ns = 90ns$$

$$T_{\mathbf{P2}} = 20 * 2ns = 40ns$$

$$T_{\mathbf{P3}} = 17 * 2ns = 34ns$$

Question 2.3

$$S_{\mathbf{P1}}^{\mathbf{P3}} = 90ns/34ns \simeq 2.65$$

$$S_{\mathbf{P2}}^{\mathbf{P3}} = 40ns/34ns \simeq 1.18$$

... continued on the next page

Problem 3

Question 3.1

The execution time at 4.0 GHz for the binary64 operations is

$$t_{b64}(@4.0\ GHz) = 10^9/(4 \cdot 10^9) = 0.25 \quad [s]$$

For the binary32 execution, two FP operations can be issued in parallel by reducing the number of cycles to

$$\text{n. cycles b32} = 10^9 \left(\frac{0.6}{2} + (1 - 0.6) \right) = 0.7 \cdot 10^9$$

Consequently,

$$t_{b32}(@4.0\ GHz) = (0.7 \cdot 10^9/(4 \cdot 10^9) = 0.175 \quad [s]$$

For the binary16 execution, four FP operations can be issued in parallel by reducing the number of cycles to

$$\text{n. cycles b16} = 10^9 \left(\frac{0.6}{4} + (1 - 0.6) \right) = 0.55 \cdot 10^9$$

Consequently,

$$t_{b16}(@4.0\ GHz) = (0.55 \cdot 10^9/(4 \cdot 10^9) = 0.1375 \quad [s]$$

Speed-up

For binary32 over binary64

$$\frac{t_{b64}}{t_{b32}} = \frac{0.25}{0.175} = 1.43$$

For binary16 over binary64

$$\frac{t_{b64}}{t_{b16}} = \frac{0.25}{0.1375} = 1.82$$

The speed-up can also be computed by the Amdahl's law

$$\text{speed-up} = \frac{1}{\frac{f_{FP}}{N} + (1 - f_{FP})}$$

... continued on the next page

Question 3.2

a) First we determine, the execution time for the binary16 execution when the clock frequency is scaled. We have:

$$t_{b16}(@3.0\ GHz) = (0.55 \cdot 10^9/(3 \cdot 10^9) = 0.183 \quad [s]$$

and

$$t_{b16}(@2.0\ GHz) = (0.55 \cdot 10^9/(2 \cdot 10^9) = 0.275 \quad [s] \quad \textbf{SLOWER}$$

To maintain a speed-up over binary64 (0.25 s), we can scale the clock to 3 GHz. To compute the power dissipation, we can use the formula:

$$P_{ave} = kV_{DD}^2 f \quad [W]$$

assuming the switched capacitance k to be constant, we determine it from the binary63 HP mode:

$$k = \frac{P_{ave}}{V_{DD}^2 f} = \frac{10}{4} = 2.5 \quad \left[\frac{W}{GHz \cdot V^2}\right]$$

Consequently, for binary16 “Medium performance" mode, we have

$$P_{b16}(@3.0\ GHz) = kV_{DD}^2 f = 2.5 \cdot (0.9)^2 \cdot 3 = 6.075 \quad [W]$$

b) The energy consumption for running the program in the two modes is:

$$E_{b64}(@4.0\ GHz) = P_{b64}(@4.0\ GHz) \times t_{b64}(@4.0\ GHz) = 10 \times 0.25 = 2.5 \quad [J]$$

and

$$E_{b16}(@4.0\ GHz) = P_{b16}(@3.0\ GHz) \times t_{b16}(@3.0\ GHz) = 6.075 \times 0.183 = 1.1 \quad [J]$$

The binary16 “Medium performance" mode is more than twice as efficient: 2.5/1.1=2.24.

... continued on the next page

Problem 4

Question 4.1

C1:

Byte offset = 2 bits (4 bytes)

Block offset = 3 bits (8 words)

Block count = $2^{12}/2^3/2^2 = 2^7$ blocks

Index = 7 bits

Tag = $32 - 7 - 3 - 2 = 20$ bits

C2:

Byte offset = 2 bits (4 bytes)

Block offset = 2 bits (4 words)

Blocks in set = 2 bits (4 blocks)

Block count = $2^{12}/2^2/2^2/2^2 = 2^6$ blocks

Index = 6 bits

Tag = $32 - 6 - 2 - 2 = 22$ bits

Question 4.2

C1:

Overhead = 2^7 blocks $*$(20 tag +1 valid) = 2688 bits

Total = $4KB * 8$+Overhead= $2^{12} * 2^3 + 2^7 * 21 = 2^7 * (256 + 21) = 35456$ bits

C2:

Overhead = 2^6 sets $*2^2$ blocks $*$(22 tag +1 valid +1 dirty +1 ref) = 6400 bits

Total = $4KB * 8$+ Overhead = $2^{12} * 2^3 + 2^6 * 2^2 * 25 = 2^8 * (128 + 25) = 39168$ bits

Question 4.3

... continued on the next page

	C1				**C2**			
Address	Tag	Index	Offset	Hit/Miss	Tag	Index	Offset	Hit/Miss
0000000000100010	0	1	2	Miss	0	2	2	Miss
0000000000100011	0	1	3	Hit	0	2	3	Hit
0000000000110100	0	1	20	Hit	0	3	4	Miss
0000010000100000	0	33	0	Miss	1	2	0	Miss
0001010000100000	1	33	0	Miss	5	2	0	Miss
0000010000100010	0	33	2	Miss	1	2	2	Hit
0001010000110000	1	33	16	Miss	5	3	0	Hit
0000000000100101	0	1	5	Hit	0	2	5	Hit

Question 4.4

$R_{\mathbf{C1}} = 3/8$

$R_{\mathbf{C2}} = 4/8$

Problem 5

Question 5.1

Solution:

Memory-stall clock cycles = Instructions/Program × Misses/Instruction × Miss penalty

Total miss cycles per instruction:

Misses/Instruction = Instruction miss rate+(Data miss rate×Data references/Instruction)

Miss penalty = 6 + Block size in words

Data references/Instruction = 40% = 0.4

The ideal CPI (no misses) is computed for Cache 1:

$$CPI_{ideal} = CPI_{measured} - CPI_{stall}(C1) = 2.00 - 0.53 = 1.47$$

Cache	miss penalty	miss cycles	miss cycles	total miss cycles	CPI
		(instr.)	(data ref.)	per instruction	
C1	6 + 1 = 7	7×4% = 0.28	7×9% = 0.63	0.28 + 0.63×0.4 = 0.53	2.00
C2	6 + 4 = 10	10×2% = 0.20	10×5% = 0.50	0.20 + 0.50×0.4 = 0.40	1.87
C3	6 + 4 = 10	10×2% = 0.20	10×3% = 0.30	0.20 + 0.30×0.4 = 0.32	1.79

C3 spends the least time on misses and C1 spend the most.

End of the Exam