End Sem Solutions
CSE 112 Computer Organization

Short answer type

Q1

a) ISA A: 5 instructions/cycle * 1000,000,000 cycle/second = 5000 MIPS [1 marks for correct calculation + 1 marks for correct answer]

b) ISA B: 6 instructions/cycle * 200,000,000 cycle/second = 1200 MIPS [1 marks for correct calculation + 1 marks for correct answer]

c) Don't know. The best compiled code for each processor may have a different number of instructions. [0.5 marks correct answer + 0.5 marks for correct explanation.]

Q2

a) R0 is **callee-saved**. We save and restore the value before using R0 and before exiting the foo function respectively. [1 marks correct answer + 1 marks for correct explanation.]

b) R1 is **caller-saved**. We use it inside the bar function without saving and restoring. [1 marks correct answer + 1 marks for correct explanation.]

c) R2's caller-callee nature is **indeterminate** on the basis of the given function. R2 was not used in either foo or bar. [1 marks correct answer + 1 marks for correct explanation.]

d) R3 is **caller-saved**. We save and restore the value of R3 just before calling bar and just after calling bar respectively. [1 marks correct answer + 1 marks for correct explanation.]

Q3

a) Total number of address in memory = $2^{(\text{address bus width})}$ = 2^5 = 32 address [1 marks correct answer]

b) Size of each memory address = Data bus width (bits) = 32 bits = 32/8 bytes =4 bytes [1 marks correct answer]

c) Total size of memory = total address x size of each memory address = 32 x 4 bytes = 128 bytes [1 marks correct answer]

Q4

a) Total number of cache lines = Size of cache/Size of one cache line = 4096/16 = 256 cache lines. [1 marks for correct calculation+ 1 marks for correct answer]

b) Number of sets in the cache cache = Total number of cache lines/Total number of ways = 256/2 = 128. [1 marks for correct calculation+ 1 marks for correct answer]

Q5

a) Average Memory Access Latency for system A = 0.9*10 + 0.1*1000 = 9 + 100 = 109 cycles [1 marks for correct calculation+ 1 marks for correct answer]

b) Average Memory Access Latency for system B = 0.95*15 + 0.05*1000 = 14.25 + 50 = 64.25 cycles [1 marks for correct calculation+ 1 marks for correct answer]

c) B would be preferred since the Average Memory Access Latency is lower for system B. [1 marks for correct answer]

Long Type Answers

Q1.

a) Total number of cache lines = Cache_size/size_of_cache_line = 512/4 = 128
[1.5 for showing the calculations + 1.5 marks for 128 = 3 marks. No partial marking in this case]

b) Cache_offset_bits = d
2^d= (size_of_cache_line/size_of_single_memory_address)
2^d= 4/1
$2^d = 2^2$
d = 2 bits
[1 mark for showing calculation + 1 mark for correct answer = 2 marks. The exact calculation steps may differ, but they should be mathematically correct.**]**

Index_bits = i
2^i= number of sets in cache
Number of sets = (cache_size/(cache_lines_in_each_set * size_of_cache_line))
(cache_lines_in_each_set for direct mapping = 1)
2^d= 512/(1*4)
$2^d = 2^7$
d = 7 bits
[1 mark for showing calculation + 1 mark for correct answer = 2 marks. The exact calculation steps may differ, but they should be mathematically correct.**]**

Tag = address_size - Index_bits - Cache_offset_bits
Tag = 32- 7 - 2
Tag = 23 bits
[1 mark for showing calculation + 1 mark for correct answer = 2 marks. The exact calculation steps may differ, but they should be mathematically correct.**]**

c) Address will be broken down in the following manner:

Tag (23 bits)	Index (7 bits)	Offset (2 bits)

32 9 2 0

[2 marks for tag + 2 marks for index + 2 marks for offset. No partial marking in this case. **]**

Q2.

There are 2 possible solution for pipeline diagram

a. **Pipeline Diagram (Decode stage stalling)**

Cycle	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
mov r1 r3	F	D	X	M	W														
mov r2 r3		F	D	X	M	W													
beq r1 r2 label1			F	D	D	D	X	M	W										
mov r3 r1				F	F	F	D	nop	nop	nop									
add r4 r5 r6							F	nop	nop	nop	nop								
label1: beq r3 r4 label2								F	D	X	M	W							
mov r6 r1									F	D	X	M	W						
add r4 r5 r6										F	D	D	D	X	M	W			
add r7 r5 r1											F	F	F	D	X	M	W		
label2: add r1 r4 r5														F	D	D	X	M	W

OR

Pipeline Diagram (Stalling in X stage)

Cycle	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
mov r1 r3	F	D	X	M	W														
mov r2 r3		F	D	X	M	W													
beq r1 r2 label1			F	D	X	X	X	M	W										
mov r3 r1				F	D	D	D	no p	no p	no p									
add r4 r5 r6					F	F	F	no p	no p	no p	no p								
label1: beq r3 r4 label2								F	D	X	M	W							
mov r6 r1									F	D	X	M	W						
add r4 r5 r6										F	D	X	X	X	M	W			
add r7 r5 r1											F	D	D	D	X	M	W		
label2: add r1 r4 r5												F	F	F	D	X	X	M	W

We get the correct value for the register after the Writeback stage. Only then the next instruction can start its execution in the next cycle. For example the dependency between instruction 2 and 3 for register R2. We stall instruction 3 till the writeback stage of instruction 2 and then only instruction 3 is executed in the next cycle.

[10 marks, one for each correct instruction. In case a student's answer is partially correct, give marks in proportion to how many instructions are correct.**]**
[2.5 marks to show the first branch is taken and 2.5 marks for second branch not taken . No partial marking for this case.**] (10 + 5 = 15 marks)**

b. **Dependencies**

RAW

1. 1 and 3 on r1
2. 1 and 4 on r1
3. 1 and 7 on r1
4. 1 and 9 on r1
5. 2 and 3 on r2
6. 4 and 6 on r3
7. 5 and 6 on r4
8. 5 and 10 on r4
9. 8 and 10 on r4
10. 7 and 8 on r6

WAW

1. 1 and 10 on r1
2. 5 and 8 on r4

WAR

1. 10 and 3 on r1
2. 10 and 4 on r1
3. 10 and 7 on r1
4. 10 and 9 on r1
5. 8 and 6 on r4
6. 7 and 5 on r6
7. 4 and 1 on r3
8. 4 and 2 on r3

[12 marks, +2 for each correct dependency. We have shown all possible dependencies. The students may write any 2 for each type.
0.5 for correct classification (WAR, RAW or WAW) + 0.5 for correct register + 1 for correct instruction number (both instruction number should be correct, else deduct 0.5 marks)= 2 mark per dependency. **]**
[-1 for every incorrect dependency. The lowest marks possible for this part is 0. Don't give less than 0 for dependencies**].**

c. Cycles required to execute the program = 19

Marking cases:

Pipeline Diagram	**Cycle Number**	**Marks**
Correct	Correct	2
Incorrect	In accordance with the pipeline diagram	1 (Partial marks for counting)

d. **Trace**

Cycle	r1	r2	r4	r6	r7
0	1	2	4	6	7
1	1	2	4	6	7
2	1	2	4	6	7
3	1	2	4	6	7
4	**3**	2	4	6	7
5	3	**3**	4	6	7
6	3	3	4	6	7
7	3	3	4	6	7
8	3	3	4	6	7
9	3	3	4	6	7
10	3	3	4	6	7
11	3	3	4	6	7
12	3	3	4	**3**	7
13	3	3	4	3	7
14	3	3	4	3	7
15	3	3	**8**	3	7
16	3	3	8	3	**8**
17	3	3	8	3	8
18	**13**	3	8	3	8

Marking:
[The trace is divided in 6 sections representing 6 changes to the registers: cycles 0 to 4, cycle 5, cycles 6 to 12, cycles 13 to 15, cycle 16 and cycles 17 to 18. All the ranges are inclusive.
Each section is worth 1 mark.
For each section, award 0.5 if the registers are modified at the end of the section and the modifications match the rubric. Award another 0.5 marks if the length of the section (in cycles) matches the rubric. **6x(0.5+0.5) = 6]**

Q3.

a) **[5x1 = 5 marks. No partial marking. 1 mark for each part]**
   a. n
   b. i-1
   c. 0
   d. i-1
   e. 2

b)
```
mov r0 $0
mov r1 $1
mov r2 $2

in r3      // Take  user input. r3 = i
mov r5 r1  // r5 = k
outerLoop: beq r3 r0 outerExit        // Exit if i == 0
           sub r3 r3 r1               // i--

           mov r4 r0                  // r4 = j
           loop1:     beq r4 r3 loop1Exit
                      out “.”
                      add r4 r4 r1
                      beq r3 r3 loop1
           loop1Exit:

           mov r4 r0                  // r4 = j
           loop2:     beq r4 r5 loop2Exit
                      out “*”
                      add r4 r4 r1
                      beq r3 r3 loop2
           loop2Exit:

           mov r4 r0                  // r4 = j
           loop3:     beq r4 r3 loop3Exit
                      out “.”
                      add r4 r4 r1
                      beq r3 r3 loop3
           loop3Exit:
           out “\n”
           add r5 r5 r2
           beq r3 r3 outerLoop
      outerExit:
```

[20 marks total. Give partial marks in proportion to how much of the code is correct.]
Note that there can be multiple solutions to this question. We have provided just one such solution.

If the student has written any instruction which is not supported, skip over the instruction in the solution, and grade accordingly.
Partial Marking criteria:

Please go through every student's solution individually, and give marks, out of 20, in proportion to the correct amount of code. If the student's code is not correct, then award marks as follows:

- If the outer loop exists: +1.5
- If the outer loop executes for n times: +1.5
- If loop1 (the loop which prints the first set of dots) exists: +1.5
- If loop1 (the loop which prints the first set of dots) executed for correct number of iterations: +1.5
- If loop1 (the loop which prints the first set of dots) prints dots: +0.5
- If loop2 (the loop which prints astrix) exists: +1.5
- If loop2 (the loop which prints astrix) executed for correct number of iterations: +1.5
- If loop2 (the loop which prints astrix) prints astrix: +0.5
- If loop3 (the loop which prints the second set of dots) exists: +1.5
- If loop3 (the loop which prints the second set of dots) executed for correct number of iterations: +1.5
- If loop3 (the loop which prints the second set of dots) prints dots: +0.5
- If the code is printing '\n': +1.5
- If the code updates k: +1.5
