Spring 2012 Name: Mark F.

Computer Architecture Test 1

Question 1. (10 points) Consider the high-level assignment statement X = A - B / C + A*B*C.

a) As in homework #1, write the LOAD and STORE assembly language instructions for this statement.

```
Load R1, A
Load R2, B
Load R3, C
DIV R4, R2, R3
MUL R5, R1, R2
MUL R5, R5, R3
SUB R6, R1, R4
ADD R6, R6, R5
STORE R6, X
```

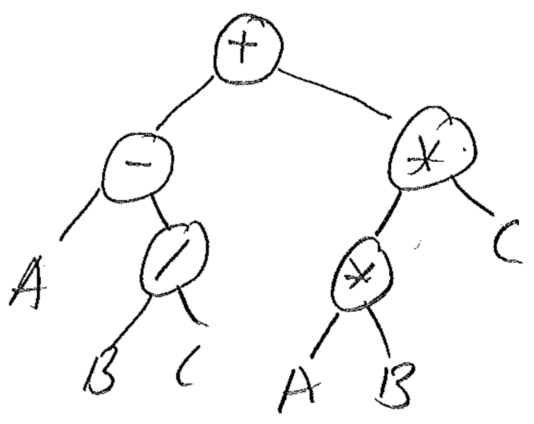
5

b) As in homework #1, write the 0-address (Stack machine) assembly language instructions for this statement.

```
PUSH A
PUSH B
PUSH C
DIV
SUB
PUSH A
PUSH B
MUL
PUSH C
MUL
ADD
POP X
```

5

10



A B C / - A B * C * +

Question 2. (13 points) Characteristics of CISC (complex instruction set computers) computers are:

- variable length instruction format
- both simple and complex instructions that require a variable number of cycles to execute
- large number of addressing modes with some complex addressing modes

Why do these characteristics make CISC computers hard to pipeline?

Time to fetch instr. varies with variable length instr.

Time to execute varies with simple & complex instructions

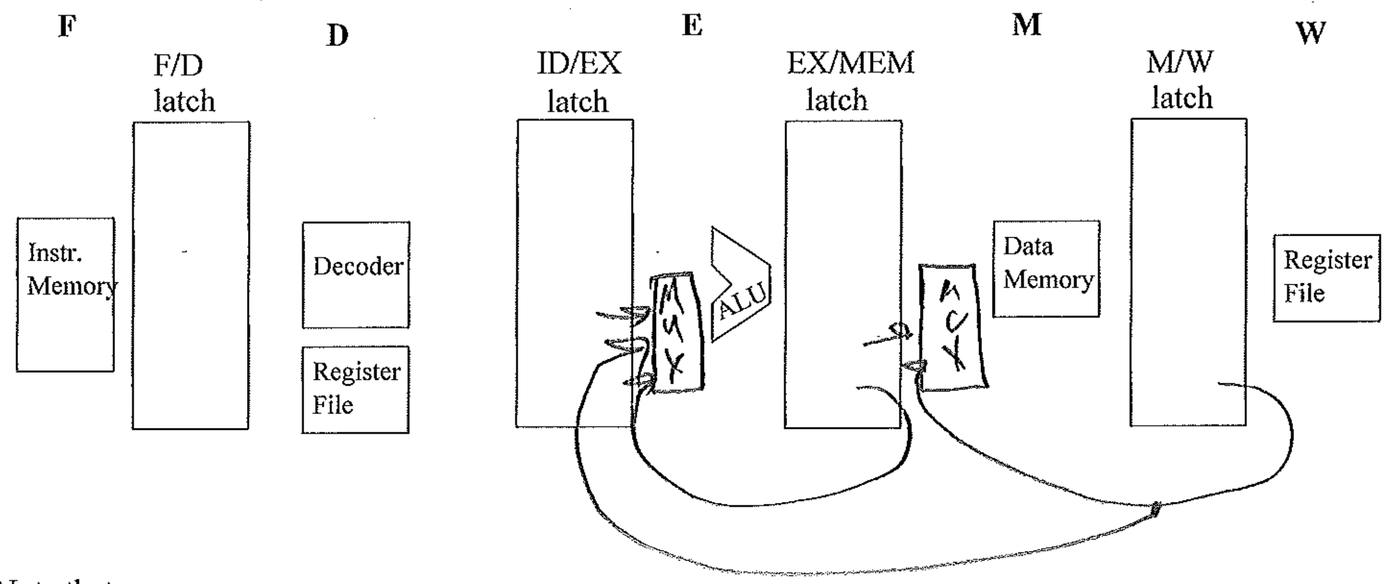
13 Complex addr. modes take longer to calculate operand addr.

Pipeline stages should be short and similar in length.

23

Spring 2012 Name:____________________

Question 3. (20 points)



5

Note that:

- The first register is the destination register, e.g., "ADD R2, R6, R7" performs R2 ← R6 + R7
- LOAD R1, 16(R2) - loads register R1 with the memory value from the address 16 + (address in R2)
- STORE R2, 8(R6) - stores register R2 to memory at the address 8 + (address in R6)

a) For the five stage pipeline of discussed in class (see above), complete the following timing diagram **assuming NO by-pass signal paths.**

7

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ADD R6, R8, R5	F	D	E	M	W													
ADD R2, R3, R4		F	D	E	M	W												
LOAD R5, 8(R3)			F	D	E	M	W											
MUL R6, R7, R2				F	–	–	D	E	M	W								
LOAD R7, 4(R6)							F	–	–	–	D	E	M	W				
STORE R7, 4(R6)											F	–	–	–	D	E	M	W

b) Complete the following timing diagram **assuming by-pass signal paths.**

8

Instructions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
ADD R6, R8, R5	F	D	E	M	W													
ADD R2, R3, R4		F	D	E	M	W												
LOAD R5, 8(R3)			F	D	E	M	W											
MUL R6, R7, R2				F	D	E	M	W										
LOAD R7, 4(R6)					F	D	E	M	W									
STORE R7, 4(R6)						F	D	E	M	W								

20

c) In the diagram at the top of the page, add all by-pass signal paths used in part (b).

Spring 2012 Name:____________________

Question 4. (12 points) Consider the conditional and unconditional branch instructions of a five-stage (F, D, E, M, W) pipelined RISC computer with 32-bit addresses.

Assembly-language Example	Machine-language Format	Description of Example Semantics
`bgt R4, R5, ELSE`	opcode | reg. # | reg. # | PC-relative displacement to label	If R4 > R5, then branch to ELSE label
`jump END_IF`	opcode | PC-relative displacement to label	Unconditionally branch to END_IF label

a) What advantage(s) does a PC-relative displacement to a label have over an absolute (32-bit) address?

- fewer # of bits

4 - relocatable - code can be moved without being modified.

b) Why is the branch penalty of a conditional branch instruction 2 cycles?

4 bgt F D E ← outcome known after E

F D } If branch is taken, need to discard

F } two instr, so penalty is 2 cycles

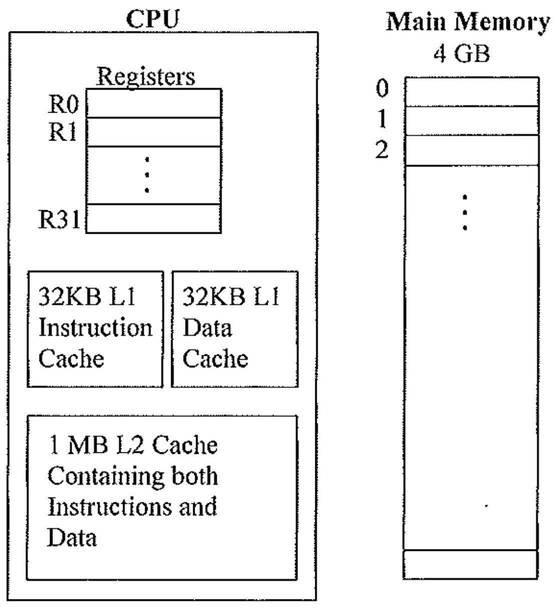
c) Why is the branch penalty of an unconditional branch instruction 1 cycle?

4 jump F D ← target of branch known

F } only need to throw away one instr.

12

Question 5. (10 points) Consider the memory hierarchy of a five-stage (F, D, E, M, W) pipelined RISC computer with 32-bit addresses below:


CPU
Registers
R0
R1
R31
32KB L1 Instruction Cache
32KB L1 Data Cache
1 MB L2 Cache Containing both Instructions and Data
Main Memory 4 GB
0
1
2

a) In a pipelined CPU, why is the L1 cache usually split into two independent caches: an instr. cache and data cache?

5 Fetch from Instr cache at same time as access data in data cache.

b) A hit ratio of 90 % in the L1 caches is common. How is this possible eventhough the program is much bigger than the L1 caches?

5 locality of reference

10

22

Question 6. (15 points) Consider the following partial program that takes a two-dimensional array M that is 100 rows x 100 columns and forms two sums:

- positiveSum - sum of all the positive values, and
- negativeSum - sum of all the negative values.

3 (a) 2 (b)

```
positiveSum = 0
negativeSum = 0
for row = 0 to 99 do                              cond. - NOT_TAKEN
   for col = 0 to 99 do                           cond. - NOT_TAKEN
      if M[row][col] > 0 then                     cond. - NOT_TAKEN or ??
         positiveSum = positiveSum + M[row][col]
      else                                        uncond.
         negativeSum = negativeSum + M[row][col]
      end if
   end for                                        uncond
end for                                           uncond.
```

3 a) Where in the code would unconditional branches be used and where would conditional branches be used?

2 b) If the compiler could statically predict by opcode for the conditional branches (i.e., select whether to use machine language statements like: "BRANCH_LE_PREDICT_NOT_TAKEN" or "BRANCH_LE_PREDICT_TAKEN"), then which conditional branches would be "PREDICT_NOT_TAKEN" and which would be "PREDICT_TAKEN"?

c) Under the below assumptions, answer the following questions.

- **all the values in M are negative**
- the five-stage pipeline from class (F, D, E, M, W)
- the target address (i.e., address of label) of all branches is known at the end of the D stage
- the outcome of conditional branches is known at the end of the E stage

i) If static predict-never-taken is used by the hardware, then what will be the total branch penalty (# cycles wasted) for the algorithm? (Here assume NO branch-prediction buffer) For partial credit, explain your answer.

4

for row	for col	if	jump end-if	end for col	end for row
2	2X100 200	2X100X100 20000	0 0	1X100X100 10000	1X100 100 =30,302

ii) If a branch-prediction buffer with one history bit per entry is used, then what will be the total branch penalty (# cycles wasted) for the algorithm? (Assume predict-not taken is used if there is no match in the branch-prediction buffer) For partial credit, explain your answer.

4

for row	for col	if	jump end-if	end for col	end for row
2	2+4x99 398	2	0	1	1 = 404

iii) If a branch-prediction buffer with two history bit per entry is used, then what will be the total branch penalty (# cycles wasted) for the algorithm? (Assume predict-not taken is used if there is no match in the branch-prediction buffer) For partial credit, explain your answer.

2

for row	for col	if	jump end-if	end for col	end for row
2	2X100	2	0	1	1 = 206

15

Question 7. (10 points) On a 32-bit computer, suppose we have a 1 GB (2^{30} bytes) memory that is byte addressable, and has a 4 MB (2^{22} bytes) cache with 16 (2^4) bytes per block.

2 a) How many total lines are in the cache? $\frac{2^{22}}{2^4} = 2^{18}$ lines

1 b) If the cache is direct-mapped, how many cache lines could a specific memory block be mapped? 1

2 c) If the cache is direct-mapped, what would be the format (tag bits, cache line bits, block offset bits) of the address? (Clearly indicate the number of bits in each)

10	18	4
tag	line #	offset

1 d) If the cache is 4-way set associative, how many cache lines could a specific memory block be mapped to? 4

2 e) If the cache is 4-way set associative, how many sets would there be? $\frac{2^{18}}{2^2} = 2^{16}$ sets

2 f) If the cache is 4-way set associative, what would be the format of the address?

12	16	4
tag	set #	offset

$\frac{2}{10}$

Question 8. (5 points) Why are full-associative caches limited to a small number (8 to 64) of cache lines?

cost of comparitors for each cache line to compare tags and search bit string of results for hit would be too slow for large cache.

5

Question 9. (5 points) The format of a memory address using a direct-mapped cache is:

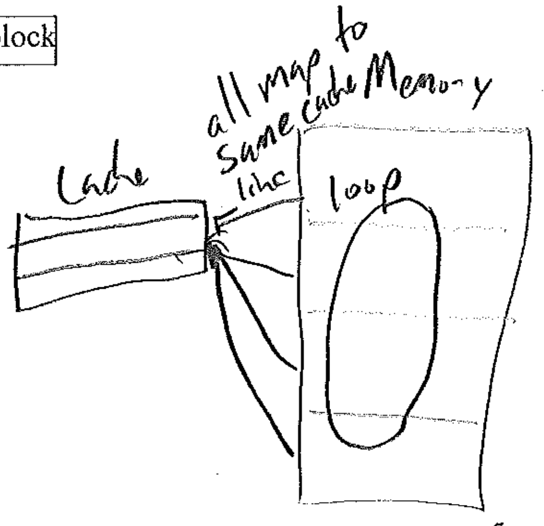
tag bits	line #	offset in block

Why would swapping the order of the tag bits and line # fields:

line #	tag bits	offset in block

5 give really bad perform of the cache?

Since the line # is the most significant bits, sequential blocks would map to same cache line causing constant misses with sequential acces



20