

## Computer Architecture Test 1

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

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

b) As in homework #1, write the 3-address assembly language instructions for this statement.

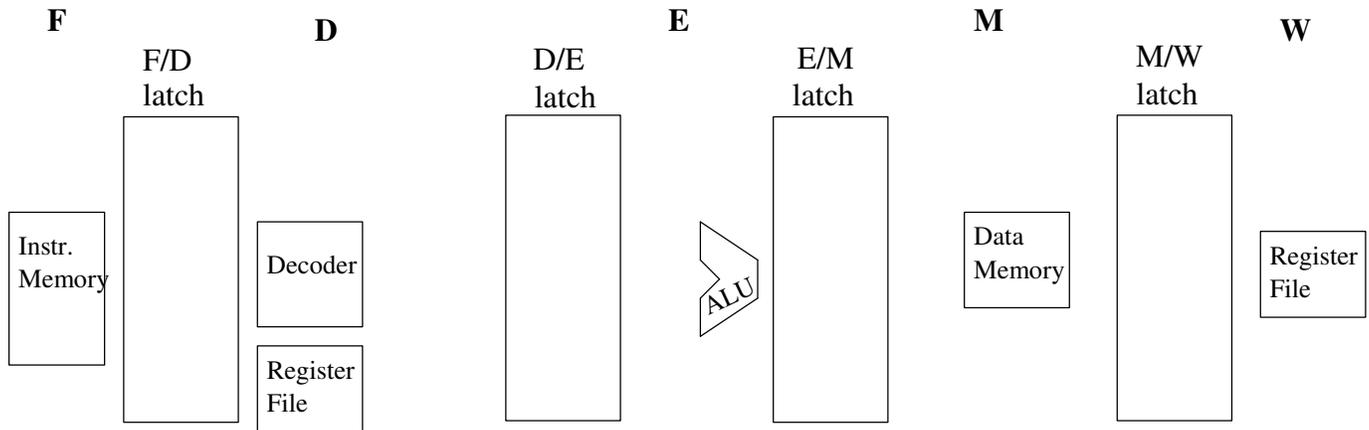
Question 2. (15 points) How does each of the following RISC (reduced instruction set computers) characteristics aid in an instruction pipeline?

a) fixed-length instruction formats (e.g., all instructions 4 bytes)

b) simple addressing modes

c) register-to-register operations of a LOAD/STORE machine

Question 3. (25 points)



Note that:

- The first register is the destination register, e.g., "ADD R2, R6, R7" performs  $R2 \leftarrow R6 + R7$
- LOAD R1, 16(R2) - loads the value from memory at the address  $16 + (\text{address in } R2)$  into R1
- STORE R2, 8(R6) - stores the value in R2 to memory at the address  $8 + (\text{address in } R6)$

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

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

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

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

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

Question 4. (25 points) Consider the following sequential search algorithm that searches an array for a specified "target" value. The index of where the "target" value is found is returned. If the "target" value is not in the array, then -1 is returned.

```
SequentialSearch (integer numberOfElements, integer target, integer array numbers[]) returns an integer
integer test;
for test = 0 to (numberOfElements-1) do
    if number[test] == target then
        return test;
    end if
end for
return -1;
end SequentialSearch
```

- a) Where in the code would unconditional branches be used and where would conditional branches be used?
- 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.
- numberOfElements = 100 and the "target" is **not** found in the array (i.e., an unsuccessful search)
  - the five-stage pipeline from class (F, D, E, M, W)
  - the target address (address of label being jumped to) 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.
- 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.
- 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.

Question 5. (15 points) On a 32-bit computer, suppose we have a 2 GB ( $2^{31}$  bytes) memory that is byte addressable, and has a 1 MB ( $2^{20}$  bytes) cache with 32 ( $2^5$ ) bytes per block.

- a) How many total lines are in the cache?
- b) If the cache is direct-mapped, how many cache lines could a specific memory block be mapped to?
- 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)
- d) If the cache is fully-associative, how many cache lines could a specific memory block be mapped to?
- e) If the cache is fully-associative, what would be the format of the address?
- f) If the cache is 8-way set associative, how many cache lines could a specific memory block be mapped to?
- g) If the cache is 8-way set associative, how many sets would there be?
- h) If the cache is 8-way set associative, what would be the format of the address?

Question 6. (10 points) Consider the results of running two programs (A and B) on identical processors, except for their cache types:

Cache Type	Program A's Execution Time	Program B's Execution Time
Direct-mapped cache	20 seconds	300 seconds
2-way set-associative cache	15 seconds	302 seconds
4-way set-associative cache	14 seconds	305 seconds

- a) For program A, explain why changing from a direct-mapped cache to a 2-way set-associative cache improved performance.
- b) For program B, explain why changing from a direct-mapped cache to a 2-way set-associative cache did not improve performance.