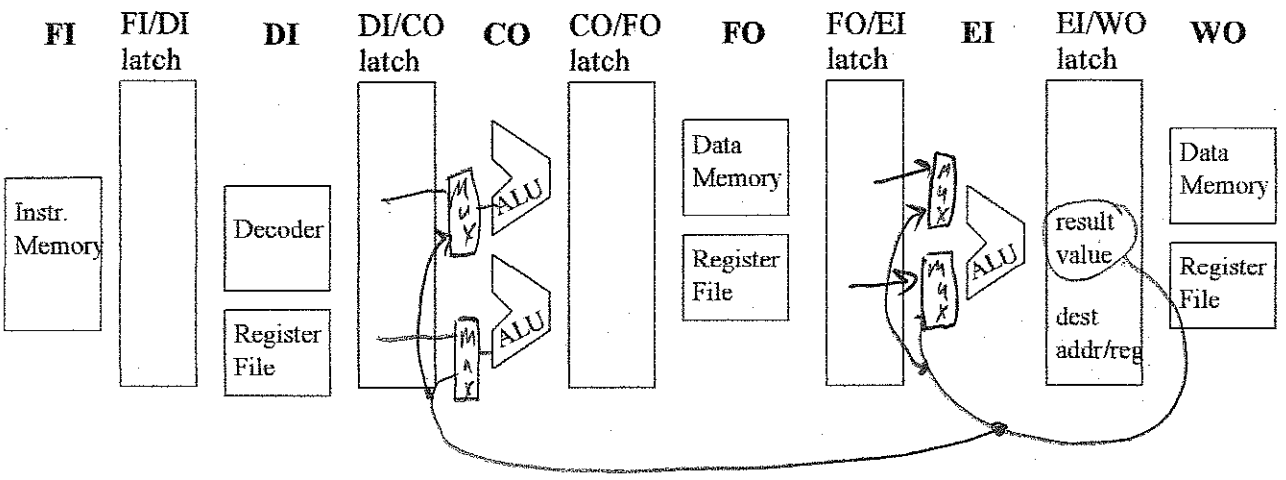
Fall 2009 Name: Mark F.

Computer Architecture Test 1

Question 1. (25 points)



6

a) For the six stage pipeline of the text (see above), complete the following timing diagram **assuming NO by-pass signal paths**. (RECALL that LOAD reads a memory value into a register; STORE saves a register to memory) Assume that we cannot write to a register and read from that same register in the same clock cycle.

Instructions	Time → 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
I1: LOAD R5, 8(R6)	FI	DI	CO	FO	EI	WO																
I2: ADD R2, R3, R5		FI	DI	CO	–	–	FO	EI	WO													
I3: SUB R5, R2, R3			FI	DI	–	–	CO	–	–	FO	EI	WO										
I4: STORE R5, 4(R6)				FI	–	–	DI	–	–	CO	+	–	FO	EI	WO							
I5: LOAD R8, 12(R5)							FI	–	–	–	–	–	DI	CO	FO	EI	WO					
I6: STORE R8, 0(R3)													FI	DI	CO	–	–	FO	EI	WO		

8

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	19	20	21	22
I1: LOAD R5, 8(R6)	FI	DI	CO	FO	EI	WO																
I2: ADD R2, R3, R5		FI	DI	CO	FO	EI	WO															
I3: SUB R5, R2, R3			FI	DI	CO	FO	EI	WO														
I4: STORE R5, 4(R6)				FI	DI	CO	FO	EI	WO													
I5: LOAD R8, 12(R5)					FI	DI	–	CO	FO	EI	WO											
I6: STORE R8, 0(R3)						FI	–	DI	CO	FO	EI	WO										

8

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

d) For the above program, indicate pairs of instruction that have **(one pair is enough for each type)**

i) write-read/read-after-write (RAW)/"true" data dependencies - R5 between I1 and I2

3

ii) output/write-write/write-after-write (WAW) dependencies - R5 between I1 and I3

iii) antidependencies/read-write/write-after-read (WAR) dependencies - R5 between I2 and I3

25

Question 2. (25 points) Consider the following code segment that counts the number of zero entries in a two-dimensional array named "matrix."

```
count = 0
for row = 1 to n do                                  <- cond
    for column = 1 to n do                           <- cond
        if matrix[row][column] == 0 then             <- cond
            count = count + 1
        end if
    end for                                          <- uncond
end for                                              <- uncond
```

PREDICT_NOT_TAKEN
PREDICT_NOT_TAKEN
PREDICT_TAKEN

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

3 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:

- n = 100, i.e., matrix is a 100 x 100
- there are 50 elements in the matrix that are 0 with none of the 0 elements adjacent on a row
- the outcome of conditional branches is known at the end of the EI stage
- target addresses of all branches is known at the end of the CO stage

Under the above assumptions, answer the following questions:

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-history table) For partial credit, explain your answer.

8

for row cond.	end for row uncond	for column cond	end for column uncond	if cond.
4	2X100	4 x100	2x100x100	4X(100X100-50)

ii) If a branch-history table 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-history table) For partial credit, explain your answer.

4

for row cond	end for row uncond	for column cond	end for column uncond	if cond
4	2	8 x100-4	2	8x50

iii) Explain how a branch-history table with two history bits per entry (i.e., two wrong predictions needed before changing the prediction) would decrease the total branch penalty (# cycles wasted) for the algorithm? (Assume predict-not taken is used if there is no match in the branch-history table)

4

for row cond	end for	for column cond	end for	if cond
4	2	4x100	2	4x50

25

Question 3. (15 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 hard to pipeline?

Pipelining works best if each stage can be short and the same length regardless of what type of instr. is being executed.

- Variable length instr. format makes it hard to know how much memory to read during the fetch to get a whole instr.
- complex instrs take longer to execute stalling instr. behind them.
- complex addr. modes make fetching operands inconsist w.r.t. time to calculate the effective address.

Question 4. (10 points) Characteristics of RISC (reduced instruction set computers) computers are:

- only LOAD and STORE instructions access memory using simple addressing modes
- arithmetic and conditional branch instructions only use register operands
- large number of registers
- fixed length (e.g., all 32-bit) instructions

Why is a large number of registers important for keeping the amount of CPU to/from memory traffic low?

Many reg.s allow data to be loaded once and used many times to avoid reloading it a gain. later.

Question 5. (15 points) Assume ADD and SUB take one cycle, LOAD takes six cycles, and MUL takes four cycles to execute.

a) What would be the first instruction to complete using Tomasula's algorithm on the following program?

```
1 LOAD  R4, 4(R8)
2 MUL   R2, R4, R7
3 STORE R6, 8(R2)
4 SUB   R3, R2, R4
5 ADD   R2, R10, R12   <- Started at 5 and finish at 6 First to complete.
6 LOAD  R12, 16(R2)
  ADD   R1, R2, R3
```

b) How does register renaming help the STORE instruction save to the correct memory address while still allowing later instructions to execute?

The STORE will be waiting on the Reservation Station for the MUL to supply it a value for the effective address. Thus, the ADD instr. can execute without effecting the STORE.

Question 6. (10 points) The Intel x86 family of processors (including the Pentium IV discussed in class) starting in the early 70's. Since the idea of RISC had not been thought of yet, the x86 instruction set is a CISC. Explain how the more modern processors (like the Pentium IV) in this family are able to make use of RISC concepts while still executing the x86 CISC programs.

The x86 CISC instructions are dynamically translated to RISC-like micro-operations so they can be "pipelined + supersalared."