McGill University
ECSE 425
COMPUTER ORGANIZATION AND ARCHITECTURE
Fall 2009
Midterm Examination

SOLUTIONS

3:30 PM – 5:00 PM, October 16th, 2009

Duration: 90 minutes

Question 1. Short Answers (20 points).
There are 10 sub questions (2 points each).

For each question below, provide a short answer in 1-2 sentences.

(a) What factors contribute to the pipelining overhead?
Propagation delay and setup time of pipeline registers, and clock skews.

(b) What is the impact of pipelining overhead?
Overhead makes the pipelined clock cycle longer.

(c) Suppose that the CPU and the memory are two critical parts of your new system. The CPU has a MTTF of a, and the memory has a MTTF of b. Assume that each fails independently of the other. What is the overall system MTTF?

$$\text{MTTF} = \frac{1}{\frac{1}{a} + \frac{1}{b}}$$

(d) Why using ratio is a good way to compare computer performance?
Because performance comparison using ratio does not depend on the reference computer.

(e) What is the difference between true dependency and name dependency?
True dependency has an actual data flow between the two dependent operands. Name dependency has no data flow, only the names are the same.

For the next 3 sub-questions, use the following code segment:

```
DIV.D    F0, F2, F4
ADD.D    F6, F0, F8
S.D      F6, 0(R1)
SUB.D    F8, F10, F14
MUL.D    F6, F10, F8
```

Name: **ID:**

Identify one of each of the following hazards (name the instructions and the dependent registers)

(f) RAW hazard:
F0 in I1 and I2, F6 in I2 and I3, F8 in I4 and I5

(g) WAR hazard:
F8 in I2 and I4, F6 in I3 and I5

(h) WAW hazard:
F6 in I5 and I2,

(i) In order to preserve data flow, what kinds of dependency need to be maintained?
Both data dependency and control dependency.

(j) In precise exception, are the exceptions handled as they occurred? If not, in what order are they handled?
No. Exceptions are handled in the order of the instructions, not as the exceptions occur.

Question 2. Performance (20 points):

There are two sub questions

(a) (10 points) Assume a four-stage pipeline where the branches are resolved at the end of the second cycle for unconditional branches and at the end of the third cycle for conditional branches. Suppose that 20% of all instructions are conditional branches (60% are taken) and 5% are unconditional branches or procedure calls. Ignore all other pipeline stalls.

1. What is the CPI for this computer if we don't use any branch prediction?

Number of stalls for conditional branches is 2 cycles.
Penalty for conditional branches $= 0.2 \times 2$

Number of stalls for unconditional branches is 1 cycle.
Penalty for unconditional branches $= 0.05 \times 1$

CPI$=1 + 0.2 \times 2 + 0.05 \times 1 = 1.45$

2. What is the CPI for this computer if the compiler predicts all branches as taken?

Number of stalls for taken branches is 1 cycle because one cycle is required to calculate the branch target address.
Penalty for taken branches $= 0.2 \times 0.6 \times 1$

Number of stalls for untaken branches is 2 cycles.
Penalty for untaken branches $= 0.2 \times 0.4 \times 2$

Number of stalls for unconditional branches is 1 cycle.
Penalty for unconditional branches $= 0.05 \times 1$

$$\text{CPI} = 1 + 0.2 \times 0.6 \times 1 + 0.2 \times 0.4 \times 2 + 0.05 \times 1 = 1.33$$

(b) (10 points) Your company has just bought a new dual Pentium processor to replace the single core one, and you are tasked with optimizing your software for this processor. You will run two applications on this dual Pentium with unequal resource requirements. The first application needs 80% of the resources, and the other only 20% of the resources. Assume that 40% of the first application is parallelizable while the second one is non-parallelizable.

1. How much speedup would you achieve if you only ran the first application?

Speedup=

$$\frac{1}{0.4 \times 0.5 + 0.6} = 1.25$$

2. How much *overall system speedup* would you observe when running both applications?

First approach: Run 80% of application 1 and application 2 in serial on one core and 20% of application 1 on the other core.

Speedup=

$$\frac{1}{0.4 \times 0.8 \times 0.5 + (1 - 0.4 \times 0.8)} = 1.1905$$

Second approach: Run 80% of application 1 on one core and 20% of application 1 and application 2 in serial on the other core.

Speedup=

$$\frac{1}{\frac{0.4 \times 0.8}{2} + \frac{0.2 + 0.2}{2} + (1 - (0.8 \times 0.4) - (0.2 + 0.2))} = 1.5625$$

Question 3. Integer Pipelining (30 points).
There are two parts to this question.

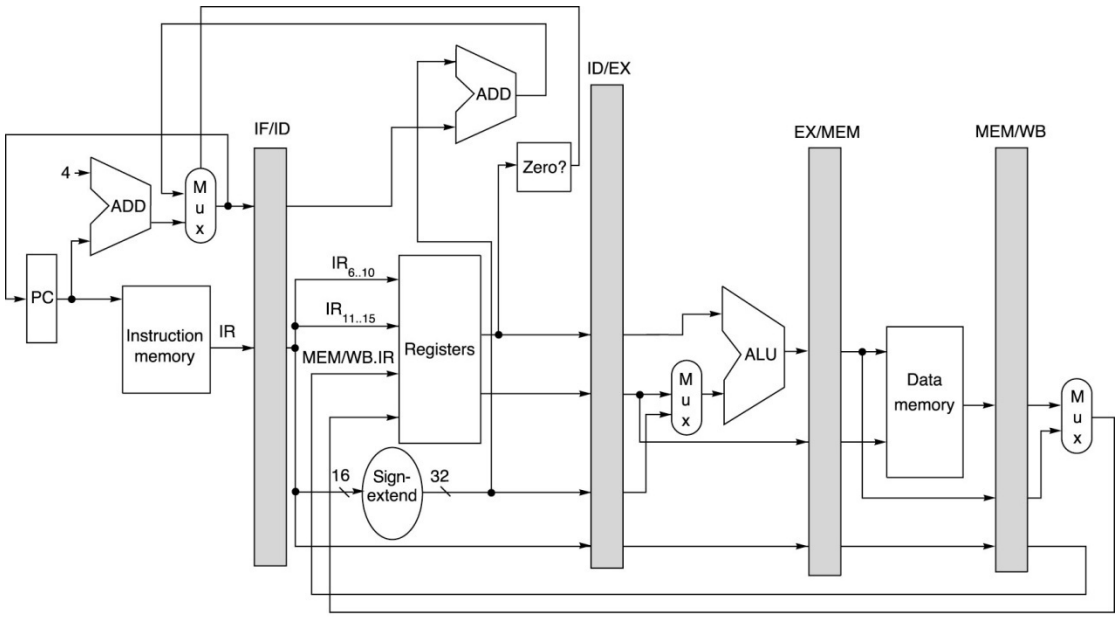
(a) (12 points) Consider the following code fragment which implements the integer code x[i] = 2x[i]+1;

```
Loop:   LD      R2,0(R1)
        DMUL    R2,R2,#2
        DADDI   R2,R2,#1
        SD      R2,0(R1)
        DADDI   R1,R1,#8
```

Name: **ID:**

```
SUB     R4,R3,R1
BNEZ    R4,Loop
```

The code will run on the classic 5-stage (I, D, E, M, W) RISC pipeline shown in the figure below. Assume all memory accesses take 1 clock cycle.



Fill in the chart on the next page to show the timing of this instruction sequence **without** any forwarding or bypassing hardware but assuming a register read and a register write in the same clock cycle. Assume that the branch is handled by flushing the pipeline and branches have a one-cycle delay.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LD	I	D	E	M	W																	
DMUL		I	D	S	S	E	M	W														
DADDI			I	S	S	D	S	S	E	M	W											
SD						I	S	S	D	S	S	E	M	W								
DADDI									I	S	S	D	E	M	W							
SUB												I	D	S	S	E	M	W				
BNEZ													I	S	S	D	S	D	E	M	W	W
Delay																I	-	-	-			-
target																			I			D

Name: **ID:**

Assume that the initial value of R3 is R1+72. How many clock cycle does the whole loop take to execute?

The branch is taken 8 times and each iteration takes 18 cycles, the last iteration the branch is untaken so it takes 21 cycles till the write back stage of the branch.

As a result the total cycles are$18 \times 8 + 21 = 165$.

(b) **(18 points)** Consider the same code fragment:

Assume the branch is handled by a one-cycle delayed branch. Show the scheduled code with delayed branch.

```
Loop:   LD      R2,0(R1)
        DMUL    R2,R2,#2
        DADDI   R2,R2,#1
        DADDI   R1,R1,#8
        SUB     R4,R3,R1
        BNEZ    R4,Loop
        SD      R2,-8(R1)
```

Assume now that the pipeline has full hazard detection and forwarding hardware (the forwarding paths are not shown). Assume a register read and a register write in the same clock cycle "forward" through the register file. Assume all memory accesses take 1 clock cycle.

Show the timing of the **scheduled code fragment with delayed branch** (derived above) for the RISC pipeline**.** Fill in the chart below to show the timing. Add arrows to indicate any forwarding. The first line is filled in for you.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
LD	I	D	E	M	W											
DMUL		I	D	S	E	M	W									
DADDI			I	S	D	E	M	W								
DADDI					I	D	E	M	W							
SUB						I	D	E	M	W						
BNEZ							I	S	D	E	M	W				
SD									I	D	E	M	W			
target										I	D	E	M	W		

Assume the initial value of R3 as R1+72, how many clock cycles does this loop take to execute? Execution Cycles=$9(cycles\ per\ iteration) \times 8(iterations) + 13(last\ iteration) = 85$

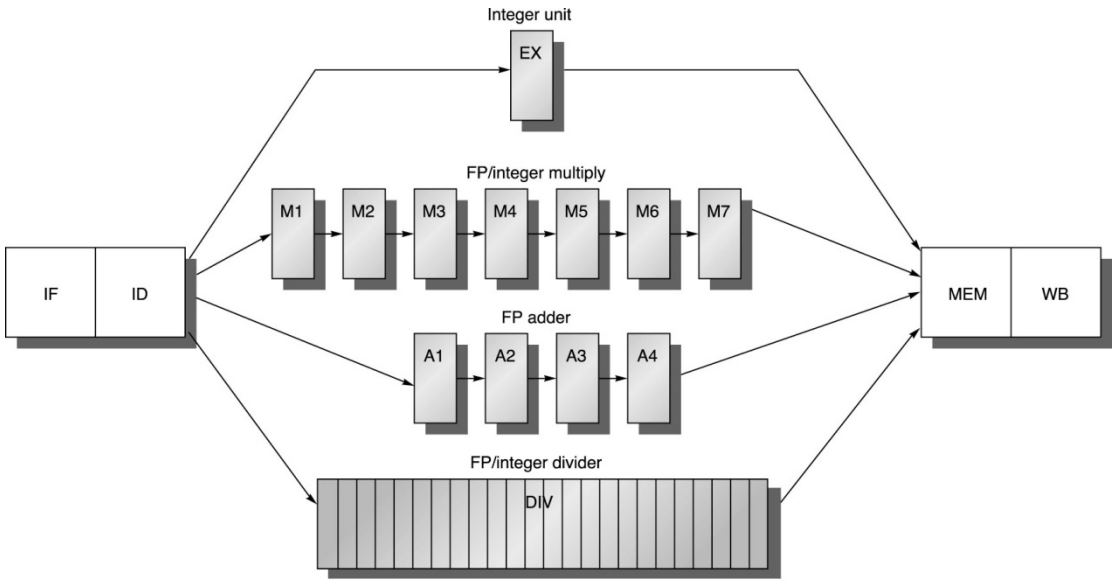
Name: **ID:**

Question 4. FP Pipelining and Loop Unrolling (30 points).
There are 2 sub questions.

Consider the following loop which computes Y[i]=a × X[i] + Y[i], the key step in Gaussian elimination.

```
loop:   L.D      F0, 0(R1)
        MULT.D   F0, F0, F2
        L.D      F4, 0(R2)
        ADD.D    F0, F0, F4
        S.D      F0, 0(R2)
        DADDUI   R1, R1, #-8
        DADDUI   R2, R2, #-8
        BNEZ     R1, loop
```

The code will run on the standard MIPS FP pipeline shown in the figure below.



(a) (**15 points)** Assume that the pipeline has full hazard detection and forwarding hardware. Assume a register read and a write in the same clock cycle “forwards” through the register file. Assume all memory accesses take 1 clock cycle. Assume that the branch is handled by flushing the pipeline. Fill in the chart on the next page to show the timing of 1 loop iteration.

Name: **ID:**

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
LD	I	D	E	M	W																	
MULT		I	D	S	M1	M2	M3	M4	M5	M6	M7	M	W									
LD			I	S	D	E	M	W														
ADD					I	D	S	S	S	S	S	A1	A2	A3	A4	M	W					
SD						I	S	S	S	S	S	D	S	S	S	E	M	W				
ADDUI												I	S	S	S	D	E	M	W			
DADUI																I	D	E	M	W		
BNEZ																	I	D	E	M	W	
flush																		I	-			
target																			I	D	E	M

How many clock cycles does a single iteration of this loop take? Since the target instruction executes in iteration 19, a single loop of the code takes 18 cycles.

(b) Here is the code fragment again

```
loop:   L.D      F0, 0(R1)
        MULT.D   F0, F0, F2
        L.D      F4, 0(R2)
        ADD.D    F0, F0, F4
        S.D      F0, 0(R2)
        DADDUI   R1, R1, #-8
        DADDUI   R2, R2, #-8
        BNEZ     R1, loop
```

Assume the pipeline latencies and initiation intervals as in Table 1. Unroll the loop as many times as necessary to schedule it without any delays. Assume the branch is now handled by a one-cycle delayed branch. Show the unrolled loop. How many clock cycles does it take per original loop iteration?

Functional unit	Latency	Initiation interval
Integer ALU	0	1
FP add	3	1
FP multiply	6	1
FP divide	24	25

Table 1. Latency and Initiation Interval of FP unit

If we unroll the loop 3 times it will take 18 cycles according to the following rescheduled:

```
LD      F0,0(R1)
LD      F8,-8(R1)
LD      F16,-16(R1)
MULT    F0,F0,F2
MULT    F8,F8,F2
MULT    F16,F16,F2
LD      F4,0(R2)
LD      F12,-8(R2)
LD      F20,-16(R2)
DADDUI  R1,R1,# -24
ADD.D   F0,F0,F4
ADD.D   F8,F8,F12
ADD.D   F16,F16,F20
DADDUI  R2,R2,# -24
SD      F0,24(R2)
SD      F8,16(R2)
BNEZ    R1,loop
SD      F16,8(R2)
```