Assignment 3
Due: Tuesday, November 19, 2013

- **Show all your work clearly and legibly for all your answers.**
- You are required to answer first 9 questions. You may provide your answer to the 10th question, which is not for marking.
- The assignment (written or printed) must be stapled before handing in.
- You may hand in your assignment
  - to the instructor on due date by end of the class or
  - deposit it on hand-in lockers on 4th floor of the E2 building besides the elevator by 430 pm on the due date
- The numbers under [ ] indicate the marks allocated to each question.

---

1. [10] A program segment has four instructions and the following table summarizes the time (in picoseconds) required by various components in a processor.

Instruction	Inst. Mem	Reg Read	ALU	Data Mem	Reg Write	Total time
lw	100	50	100	100	50	400
sw	100	50	100	100		350
add	100	50	100		50	300
beq	100	50	100			250

a. What is the speed up you would expect running the program in a pipelined processor over non-pipelined processor?
b. If the ALU unit takes double the time (ie 200ps), what is the speedup you would expect running the program in a pipelined processor over a non-pipelined processor?

Solution:

a) In a non-pipelined implementation the cycle time is determined by the slowest (longer execution time) instruction, which is the *lw* in this case. Therefore, the cycle time is 400ps. This program segment when run on a non-pipelined processor, will require 4

cycles to complete. Therefore, the execution time = 4*400=1600ps

In a pipelined implementation, the cycle time is determined by the slowest component of the datapath, which is ALU or IM in this case at 100ps. The number of cycles required is 4 cycles and the execution time =4*100=400ps.

Speed up = 1600/400 =4

b) new ALU time =200 ps

In a non-pipelined implementation the cycle time is determined by the slowest (longer execution time) instruction, which is the *lw* in this case. Therefore, the cycle time is 500ps. This program segment when run on a non-pipelined processor, will require 4 cycles to complete. Therefore, the execution time = 4*500=2000ps

In a pipelined implementation, the cycle time is determined by the slowest component of the datapath, which is ALU in this case at 200ps. The number of cycles required is 4 cycles and the execution time =4*200=800ps.

Speedup = 2000/ 800=5/2=2.5

2. [6] Various stages of the datapath have latencies summarized in the following table.
   a. For this implementation, what is the clock cycle time in a pipelined and non-pipelined processor?
   b. For this implementation, what is the total latency of an "*lw*" (load word) instruction in a pipelined and non-pipelined processor?
   c. If we can split one of the stages of the pipelined datapath into two new stages, each with half the latency o the original stage, which stage would you split and what is the new clock cycle time of the processor?

IF	ID	EX	MEM	WB
200	300	100	250	150

Solution:

a) Pipelined clock cycle time: 300ps; non-pipelined clock cycle time: 1000ps
b) Pipelined latency of lw instruction: 5*300=1500ps; non-pipelined latency: 1000ps
c) The ID stage would be split into two half each taking 150ps; The new clock cycle time for this datapath is 250ps

3. [10] For the following sequence of instructions
   a. [2] Indicate the dependences and their type (as Read After Write (RAW) or Write After Read (WAR) or Write After Write (WAW))
   b. [4] Assume there is no forwarding in this pipelined processor. Indicate hazards and add NOP instructions to eliminate them.
   c. [4] Assume there is full forwarding. Indicate hazards and add NOP instructions to eliminate them

```
sw $16, -100($6)
lw $4, 8($16)
add $5, $4, $4
```

Solution:

a) Between instruction 2 and instruction 3 there is a RAW dependency
b) Delay the *add* instruction to avoid RAW hazard on $4 from *lw* instruction

```
sw $16, -100($6)
lw $4, 8($16)
NOP
NOP
add $5, $4, $4
```

c) With full forwarding the ALU can forward a value to the EX stage of the next instruction without a hazard. However, a *lw* cannot forward to the EX stage of the next instruction (but can to the instruction after that). The code that eliminates these hazards by inserting NOP instructions is:

```
sw $16, -100($6)
lw $4, 8($16)
NOP
add $5, $4, $4
```

4. [10]
   a. [4] Assume the clock cycle time is 250ps for an implementation. What is the total execution time of the sequence of instructions in question 3 in a pipelined processor without forwarding?

b. [4] Assume the clock cycle time is 300ps for an implementation. What is the total execution time of the sequence of instructions in question 3 in a pipelined processor with full forwarding?
c. [2] What is the speedup or slowdown achieved by adding full forwarding (question 4b) to a pipeline that had no forwarding (question 4a)?

Solution:

a. This is a segment of a program being executed. The total number of cycles required to complete these instructions without any forwarding would be 7+2 (nops)=9 cycles. Therefore, time required is = (7+2)* 250= 2250ps
b. With similar argument, the number of cycles required with full forwarding is 7+1=8 cycles. Therefore, the time required is = (7+1)*300=2400 ps
c. 2250/2400=0.94 (slowdown)

5. [4] Using the Graphical representation of forwarding, show the forwarding paths needed to executed the following three instructions:

```
add $2, $3, $4
add $4, $5, $6
add $5, $3, $4
```

Solution:
Instruction 3 depends on Instruction 2; there will be a forward between ALU of instruction 2 to the ALU of instruction 3

6. [5] Identify all the data dependencies in the following code. Which dependences are data hazards that will be resolved using forwarding?

```
add $2. $5, $4
add $4, $2, $5
sw $5, 100($2)
add $3, $2, $4
```

Solution:
An operand in the 2nd instruction depends on the 1st instruction (RAW)
An operand in the 3rd instruction depends on the 1st (RAW)
An operand in the 4th instruction ($4) depends on the 2nd (RAW); The $2 in the 4th instruction is not actually a dependence of $2 in 1st instruction)

Forwarding will help all

7. [5] Consider executing the following code on the pipelined datapath shown below. At the end of the 5th cycle of the execution, which registers are being read and which registers are written?

```
add $2. $3, $4
add $5, $6, $7
add $8, $9, $10
add $11, $12, $13
add $14, $15, $16
```

Solution: [5] At the end of the 5th cycle

Registers Written: $2 is written;
Note: Do not cut marks if the student mentions the pipeline registers such as PC, IF/ID, ID/EX, EX/MEM, MEM/WB in addition to $2.

Registers Read: $12 and $13 are read;

8. [10] Consider a program consisting of 100 ***lw*** instructions (consecutively) in which each instruction is dependent upon the instruction before it.
   a. [5] what would the actual CPI be if the program were run on a 5 stage pipelined processor?
   b. [5] If the slowest component in the pipelined datapath takes 500ps, what is the total time to complete this set of 100 ***lw*** instructions? (Assume that the first ***lw*** instruction completes in 2500ps)

Solution:

a) The first instruction will complete in 5 cycles. In the pipelined implementation, there will be a delay of 2 cycles between 1st and 2nd, 2nd and 3rd, 3rd and 4th and so on. There are 99 *lw* instructions following the first *lw* instruction. Therefore, total number of cycles required is

5+99*2 +99= 302
There are a total of 100 instructions, and hence CPI =302/100=3.02

b) **solution 1:** The time required to complete the first lw instruction is 2500 ps. For each additional instructions there is a delay of 2 cycles and clock cycle time of this pipelined datapath is 500ps and hence delay between each instruction is 1000ps. Therefore, the total time required to complete 100 lw instructions is:
2500+99*1000+99*500=2500+99000+49500 = 151,000ps

solution 2: Since the number of cycles required is 302 and each cycle takes 500ps total time required is (5*500+297*500) = 302*500
= **151, 000ps**

9. [20] Refer to **figure 2** for this question. In this figure the instructions in the various stages of the pipeline are not identified. Your task is
   a. to determine as much as you can about the five instructions in the five pipeline stages.
   b. If you cannot fill in a field of an instruction, state why?

For some fields Refer to slides 7,8 and 13 in lec-6c1-pipe3-Oct31 to identify the control signal values to identify the instruction. For further help, refer to slides 17 through 21 for an example of flow of a set of instructions along the pipelined datapath.

Solution:

A. IF stage [**1 mark**]: No control and data lines are labelled. So cannot say for sure what instruction is being fetched. It could be anything.
B. ID stage: **(total 6 marks)**
   a. **[1 mark]** The control bits for the stages following ID stage are EX: 1100; Mem: 000; and WB: 00
      Based on these control values the EX stage is doing a R-Type instruction. (Refer to slide 15, Notes on Oct.31 - (see below))
   b. **[1 mark]** The source registers are given in the data lines as labels: **$10 and $11**
   c. **[1 mark]** The destination register is given in the data line as label: **$1**
   d. **[3 =1+1+1 marks]** The actual operation at ALU will be one of the five operations: and, or, add, sub or slt which could be determined by the function field, the least significant 6 bits of the address that could be found based on the lower 16 bit address given as 2090 in the data label. Converting 2090 into binary **[1 mark]** and picking up the lowest 6 bits gives us 101010 which is 42 in decimal and tells us that the instruction is **slt [1 mark].** Therefore, in the ID stage the instruction is: slt $1, $10, $11 [**1 mark**]

C. EX stage **[Total 6 marks]**
   a. **[3=1+1+1 mark]** control bits in the EX Stage shows EX: 0001; Mem: 001; WB: 00; Based on these control values, this is an I-type instruction, and hence **could be a lw or sw [1 mark]**

Since the MemWrite is 1, and the WB stage control values both being 0s **suggest it is not lw**, **[1 mark]** and **hence we can say for sure that this is a sw instruction**. **[1 mark]**

b. **[1 mark]** register values are **$5 and $6**
c. **[1 mark]** lower 16 bit address **(offset value) is 16**
d. **[1 mark]** , the instruction is: **sw $6, 16 ($5)**

D. Mem stage **[Total 3 mark]**

a. **[1 mark]** The control bits in the Mem stage shows
Mem: 100 (from the table this suggests it is **beq** instruction)
WB: 00 **(**not updating a register**)**
Therefore, it is **beq** instruction.

b. **[2 mark]** only data value provided is **31**, which could not be decided since from the table in the slide 15, the RegDst is X for beq. Hence, 31 could be interpreted either as one of the source registers or the branch address. Therefore the instruction could be
b1: **beq, ____, $31, _____** **(**when RegDst is deasserted**) [2 mark]**
or
b2: beq, ____, _____, 11111?????????? (lower 16 bits) (when RegDst is asserted) **[2 mark]**

E. WB stage **[Total 3 mark]**

a. **[1 mark]** control values: 11. That is, the Register is being updated, which implies it is a lw instruction.
b. **[1 mark]** destination register is $15
c. No other data is available since they (the lower 16 bit address and source register used to compute the address in memory from which a data is read to be loaded on to the register) are already consumed at the EX stage of this instruction.
d. **[1 mark]** Therefore the instruction is
lw, $15, ?, (?)

10. **This question is not for marking**. Refer to figure 1. Instead of passing the RegWrite control signal along the datapath registers, if you send the signal to the register file right after it was generated at the control unit, discuss what happens? Are you skipping any of the assumptions of the pipelined datapath? Are any of the instructions (prior or later) affected by this change?

Figure 1: Pipelined Datapath with Control (refer this figure as see you see fit for any of the questions)

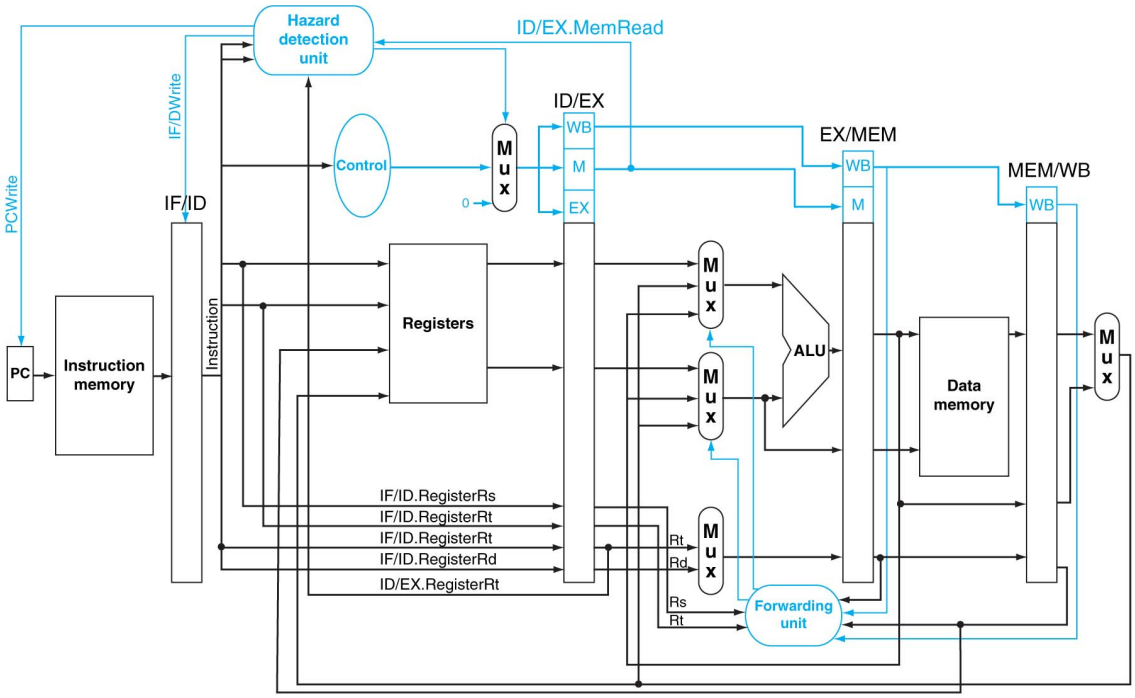

Figure 2: Pipelined datapath with five instructions (not identified)