Introduction to Computer Organization – Fall 2015

Homework 4

Assigned: Tuesday, Oct. 27, 2015
Deadline: November 10, 2015 @ 11:55 PM

Name: _____ Uniqname: _____

1. Submit a pdf of your typed or handwritten homework in CTools. Your file must be named uniqname_HW#.pdf (replacing uniqname with your uniqname and # with the assignment number).

2. Your answers should be neat, clearly marked, and concise. Computer written work is recommended (but not required). Show all your work, and state any special or non-obvious assumptions you make.

3. You may discuss your solution methods or your answers with other students, but the solutions you submit must be your own.

Problem 1

Oliver, a (fictional) 370 student, is an up and coming star in the world of computer architecture. Just after taking his midterm exam, he was approached by a representative from Know Noth Inc. to assist them with their processor design. The target program of the processor consists of 1500 LC2K instructions. After profiling the program, Oliver found its instruction mix to be:

Instruction Type	Percentage (1500 instr.)	Latency (ns)
beq	27%	13
nand	6%	22
add	12%	25
sw	15%	40
lw	35%	30
noop	5%	7

(a) At the moment, the processor is using a single cycle datapath taught in lecture. Before making any change, Oliver has to collect the baseline statistic. How many cycles are needed to complete this program? And what is the Cycles Per Instruction (CPI) for this program?

(b) If the latency of each instruction is as provided in the table above, what is the best clock period for this design?

(c) How much time does the processor use to execute the target program?

Having done phenomenally well on his midterm, Oliver knows that a single-cycle datapath is inefficient. He decides to apply his knowledge from class and implement a multi cycle LC2K datapath *as shown in lectures*.

(d) How many cycles are needed to complete this benchmark? Show your work.

(e) What is the Cycles Per Instruction (CPI) for this program?

(f) If the clock period for the multi cycle architecture is 8ns, what is the total execution time for this benchmark?

(g) Which datapath (single-cycle / multi-cycle) will this program perform best on, in terms of execution time? If we want both designs to have equal performance, which instruction's latency in the table above needs to be improved? And to what value? Show your work.

Problem 2

In this question, we will be analyzing performance of a single cycle and a multi-cycle LC2K datapath in more detail. For the following tasks, **show your work**.

(a) Consider the following LC2K program

```
            lw      0       1       one
            lw      1       2       data
            lw      0       3       data
            nand    2       2       5
            add     1       5       5
            nand    5       5       5
loop        beq     5       3       done
            lw      0       4       res
            add     4       2       4
            sw      0       4       res
            add     1       3       3
            beq     0       0       loop
done        halt
one     .fill   1
data    .fill   0
base    .fill   8
        .fill   -5
        .fill   6
res     .fill   0
```

How many cycles does it take to execute the program in the single-cycle datapath vs the multi-cycle datapath? Calculate both cycle counts, and assume *halt* takes 2 cycles in the multi-cycle datapath.

(b) In order to select the most performance-effective architecture to execute the program, it is necessary to determine the execution time on each architecture. We know the following:

Register file read/write time: 6ns
ALU operation time: 13 ns
Memory read time: 25 ns
Memory write time: 30 ns
All other operations: 0 ns

What is the minimum clock period for the single-cycle architecture? For the multi-cycle architecture?

(c) How long does it take to execute the program in the single-cycle datapath and multi-cycle datapath? Which design provides better performance based on execution time?

(d) Let's say that it is possible to split any operations listed in (b) and distribute its latency into 2 separate cycles. Which of the current architectures (single-cycle, multi-cycle, both) will benefit from this operation splitting in term of performance? For each operation that you chose to split, list its new latency per cycle. Recalculate the minimum clock period, cycle count, and execution time for each architecture that benefits from this procedure.

Problem 3

Fused Multiply Add (FMA) is an operation that computes the product of two numbers and adds it to the third number. This operation is commonly used in digital signal processing and graphical applications.

For this problem, we would like to extend the LC-2K multicycle datapath to be able to execute (simple) FMA instructions. In order to achieve this, a multiplier module is added into the system. This addition enables two new opcodes that allow us to perform regular multiplication and the FMA operation. The new opcodes are as follows:

fma <regA> <regB> <regC> // Compute ([regA] * [regB]) + [regB], the store result in regC
mul <regA> <regB> <regC> // Compute ([regA] * [regB]), the store result in regC

You will be integrating this new multiplier in to the data path. A new “Multiplier” block is added to the figure shown at the end of this problem.

(a) Connect the Multiplier to the rest of the datapath in a way that fma and mul instructions will take the least number of cycles. You are allowed to extend MUXes (i.e. from 2:1 to 4:1), but **not** allow to add any additional hardware. New input(s) to a MUX must be added **below** the bottom input of that MUX. The Multiplier block has private **internal register file** to buffer its output. Disregarding control signals, add your change on the figure and state what you did down below.

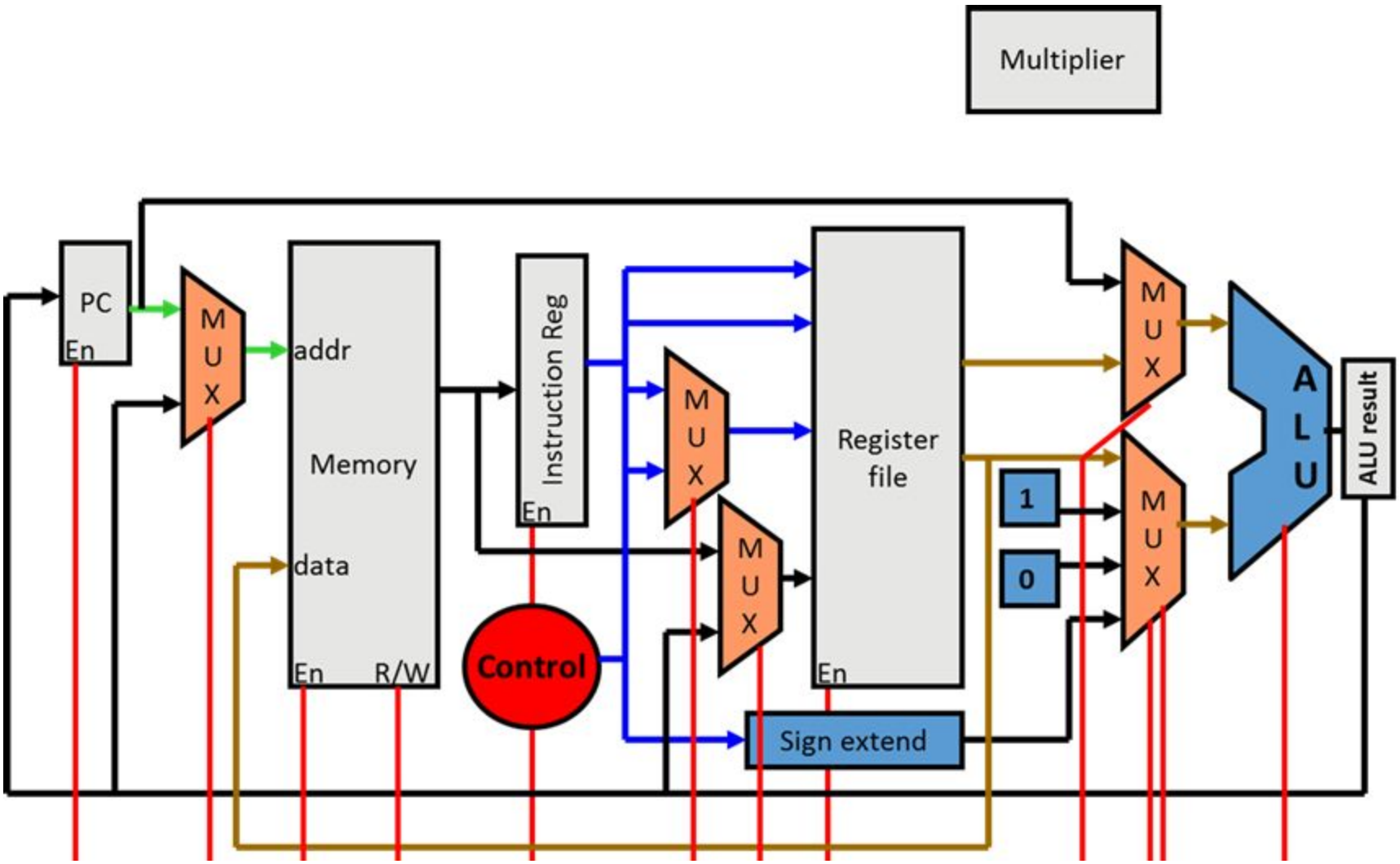
(b) How many cycles does the fma instruction take? Walk us through an execution of the fma instruction. State what happens cycle by cycle. We’ve filled in the first two cycles for you.

Cycle 1: Fetch instruction from memory.
Cycle 2: Decode the instruction opcode. Retrieve regA and RegB from the register file.
...

(c) Now we will be considering the control signals for the fma instruction. On the table below, fill in the control signal values required for the instruction to function correctly. Use x if the value of that signal does not affect the execution. Ignore any control signals needed for the Multiplier. *Note that some signals now have more than 1 bit, and that there may be more rows than required in the table below.*

fma:

Cycle	PC	Mux1 (Mem)	Mem En	Mem R/W	Instr Reg. en	Mux2 (RF in top)	Mux3 (RF in bot)	RF en	Mux4 (ALU top)	Mux5 (ALU bot)	ALU


Multiplier
PC
En
M
U
X
addr
Memory
data
En
R/W
Instruction Reg
En
M
U
X
M
U
X
Control
Register
file
En
M
U
X
1
0
M
U
X
A
L
U
ALU result
Sign extend

Problem 4

Tired from doing plumbing and saving Princess Peach, Mario decided to take EECS370 to work on another kind of pipeline. Consider the following LC-2K code executing on the 5-stage LC-2K pipeline discussed in class:

```
i0:   lw    2   3   192
i1:   add   3   3   4
i2:   nand  2   2   5
i3:   add   4   2   2
i4:   lw    4   5   136
i5:   sw    5   4   exit
```

(a) Identify Read-after-Write data dependency and data hazards presented in the above program (if any). Fill in the table provided down below. Not all rows in the table will be filled.

Instruction #	Dependent instr.	Register #	Data Hazard? (Y/N)

(b) Assuming that the pipeline is using “detect-and–stall” to resolve data hazards. Use the following table to show the stage of the pipeline that each instruction resides at each execution cycle. For the purpose of this question, assume that the register file supports read/write in the same cycle. Add * next to a stage’s name to indicate stalling. For example, if at cycle X, the processor stalls an instruction in the ID stage, fill in ID* for that table entry.

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i0:	**IF**	**ID**	**EX**	**MEM**	**WB**										
i1:															
i2:															
i3:															
i4:															
i5:															

(c) Follow the same instructions as part (b), but instead, use “detect and forward” instead of “detect and stall". The first row has been filled out for you.

Instr.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
i0:	**IF**	**ID**	**EX**	**MEM**	**WB**										
i1:															
i2:															
i3:															
i4:															
i5:															