C-355 Credit HW # 4 (Due Friday 08/25/2017 at 10 PM)

Group 1: Chris Kasper, Alfonzo Desantis, Matt Long, Dallas Foglia

Q1. - Chris Kasper

Consider the stagewise single cycle CPU with the circuit as given on the attached sheet. The following are the latencies of each component:

Instruction memory 180 ps

Add 4 unit 60 ps

Mux 15 ps

Registers 120 ps

Main Control 50 ps

ALU Control 25 ps

ALU 150 ps

AND 5 ps

Shift Left 2 10 ps (ShiftLeft2jump also)

Sign Extend 15 ps

Branch Adder 60 ps

Data Memory 150 ps

1. Do a stagewise latency analysis of the circuit. Write down signal timings on the stagewise single cycle CPU diagram provided. Reset the time to zero every time any signal crosses a stage boundary. Calculate the latency and highlight the critical path for each stage

**\*Signal timings on picture on last page\***

**Latency Analysis**

IF Stage:

Fetch instruction 180 ps (Critical Path)

PCV:=PC+4 60 ps

Latency of stage = 180 ps

ID stage;

Read registers 120 ps (Critical Path)

ShiftLeft2Jump 10 ps

SignExtend+ ShiftLeft2 25 ps

Latency of stage = 120 ps

EX stage:

Generate ALU output max{MUX,ALU Control} +ALU= max{15,25}+150 = 175 ps

Generate address of next instruction max{Adder,ALU Output+AND}+MUX+MUX =max{150,175+5}+15+15= 210 ps (Critical path)

Latency of stage = 210 ps

MEM stage

Read/Write memory 150 ps (critical path)

Latency of stage = 150 ps

WB stage

Write back register MUX + Registers = 15 +120 = 135 ps (critical path)

Latency of stage = 135 ps

1. What will be the clock rate of a pipelined CPU based on this circuit?
   1. Clock rate= max {180,120,210,150,135}= 210 ps (CCT)
      1. Clock rate= 1/210ps
2. If you were asked to increase the clock rate of the pipelined CPU by designing a faster ALU, by how much will you be able to reduce its latency beyond which it would make no difference?
   1. The next longest stage after EX is the IF stage, which is 180 ps. Therefore, if only the ALU is being redesigned to increase the clock rate, then the ALU could get reduced from 150 ps to 120 ps. Decreasing any more will make no difference because the clock rate will be based on the IF stage because it would then have the max latency of all the stages

Q2. Dallas Foglia

The table below shows the number of different instructions of each type in a given program to be run on a multicycle CPU:

Instruction Number

R-format 600

lw 300

sw 200

beq 250

j 50

1. What is the average CPI?

Assuming:

R-format = 4 cycles

lw = 5 cycles

sw = 4 cycles

beq = 3 cycles

j = 3 cycles

CPI = sum(#inst. x cycles) / total inst.

CPI = (600x4+300x5+200x4+250x3+50x3) / 1400 = 4.0

1. What is the execution time if the clock rate is 2 GHz?

Exc. Time = #inst. x CPI / clock rate

Exc. Time = 1400 x 4 / 2GHz = 2.8E-6 or 2.8 μs

1. What is the speedup obtained by 5-stage pipelining with a clock rate of 1.65 GHz? Assume there are a total of 200 hazards in the program. Out of these, 25% do not need any stalls, 60% need one stall each and the remaining need 2 stalls each

There are 180 total stalls across 1400 instructions. 120 single stalls + 30 double stalls. As opposed to 400? total stalls in the default CPU, assuming each hazard counts as 2 stalls by default.

Optimally, there are 1400 cycles, with one cycle for each instruction. 1400 cycles + the 180 cycles from the stall leaves us with 1580 total cycles.

Exc. Time = Total Cycles x Cycle period

Exc. Time = (1580 + 4) \* (1/1.65 GHz) = 9.6E-7s or 960ns

Performance = Old/New = 2.8E-6s / 9.6E-7s = 2.91 or a 291% increase in performance.

Q3 - Chris Kasper

Consider the following code:

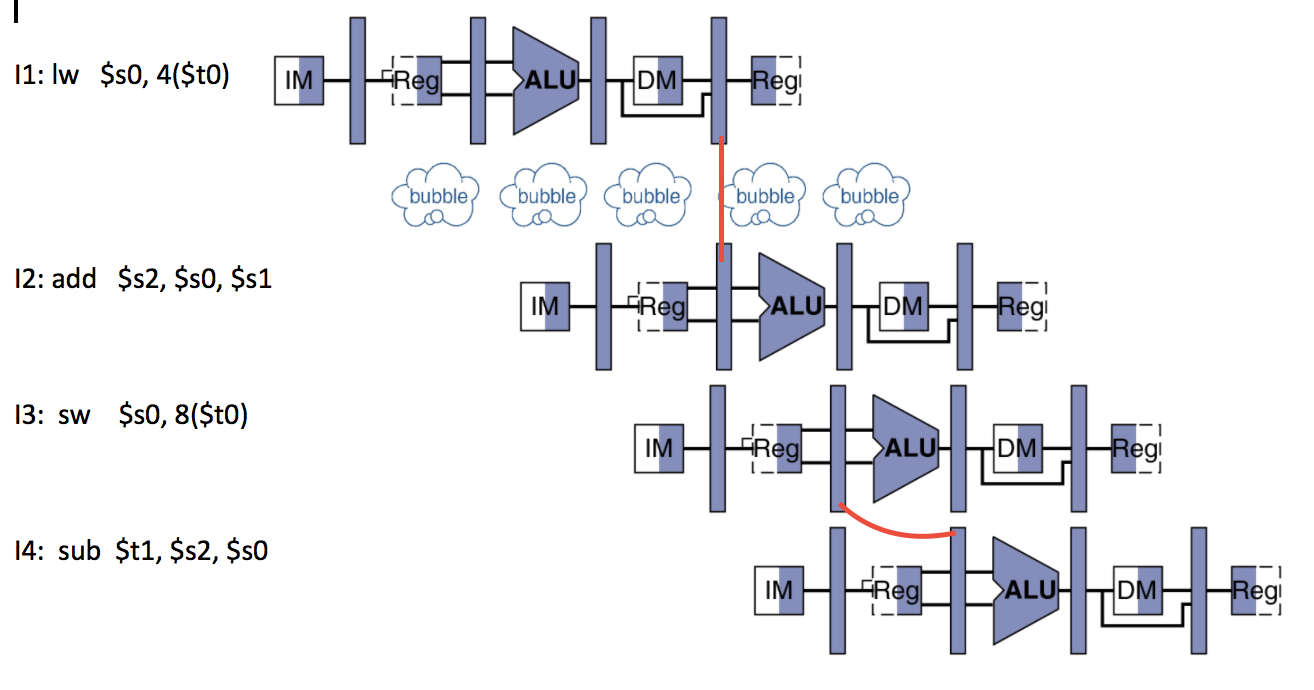
I1: lw $s0, 4($t0)

I2: add $s2, $s0, $s1

I3: sw $s0, 8($t0)

I4: sub $t1, $s2, $s0

Draw a ***stylized*** pipeline diagram showing how the code will execute assuming ***full forwarding***



Q4. -Alfonzo DeSantis

This problem deals with the pipelined datapath . Branches execute in the ID stage. Consider the following sequence of instructions:

I1: lw $s0,0($t0) (assume value read from memory is ***non-zero***)

I2: beq $s0,$zero,I5

I3: sub $s1,$s2,$s3

I4: j I6

I5: add $s1,$s2,$s3

I6 : sw $s1, 4($t0)

How many cycles does the program take to complete in each of the following cases?

In each case, clearly identify the location and number of stalls/flushes needed.

1. No forwarding, no branch prediction

I1

stall

stall

stall

I2

stall

I3

I4

Stall

I6

14 cycles (5 stalls)

1. Full forwarding, but no branch prediction

I1

Stall

I2

Stall

I3

I4

Stall

I6

12 cycles (3 stalls)

1. ALU-ALU forwarding and Always Taken predictor

I1

Stall

Stall

Stall

I2 (Predict Taken (wrong))

flush I5 do I3

I4

Stall

I6

13 cycles (4 Stalls, 1 Flush)

1. Full forwarding and dynamic predictor with 80% accuracy

I1

Stall

I2

Flush (20% of the time)

I3

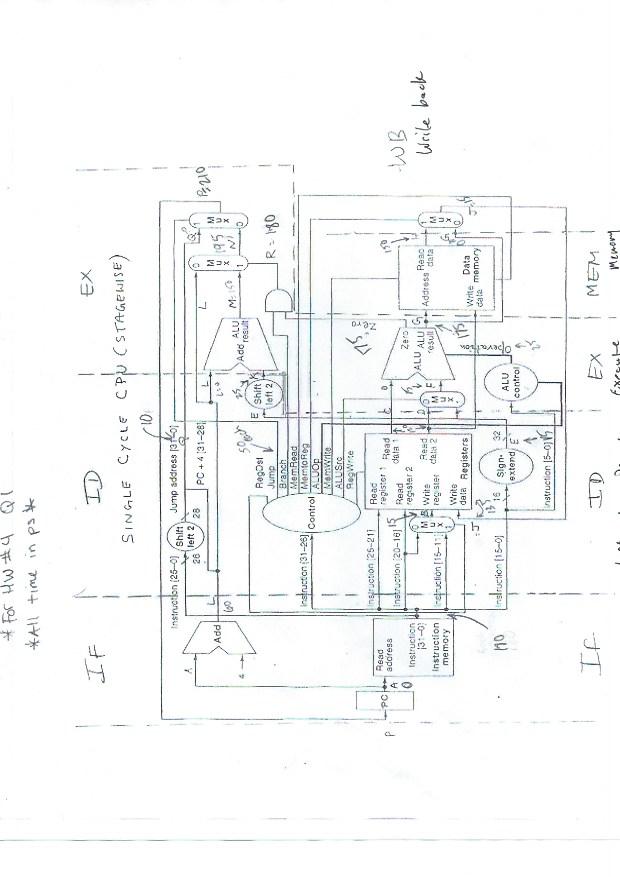
I4

Stall

I6

11.2 cycles (2 stalls, 20% flush after I2)

**Signal timings for Q1 (a)**

****