Release Date: 06/08/2020 Due Date: 06/12/2020 – 11:59pm

CSE 141 – Final Exam (100 pts)

**The total number of points that can be earned is 115, of which 15 points are extra credit.**

# 

**Submission Instructions (Read Carefully):**

1. Please type your responses in a separate document and submit the PDF file to gradescope. ONLY typed PDF documents will be accepted as a valid submission. However, if there are any questions that ask you to draw a diagram, those can be hand-drawn, but must be attached to the final PDF file. Please do not submit a separate file for hand-drawn diagrams. Only one, final PDF file will be accepted.

2. Unless you have general questions about the exam, you should not be discussing on Piazza. **If you have general questions, please email** [**bhahn221@eng.ucsd.edu**](mailto:bhahn221@eng.ucsd.edu) **or post your question to Piazza as a private question so that only instructors can see**. It is okay to refer to the class materials, but discussing with other classmates about the questions will not be allowed. It is crucial that the students do not violate the academic integrity.

3. No late submissions allowed!

4. For all questions, show your work! Answers without showing your work (derivations) may not get full credit.

# **(9 pts) Performance, Amdahl’s Law, and Pareto Optimality**

1. **(3 pts)** Suppose we have two implementations of the same instruction set architecture. Computer A has a clock cycle time of 200 ps and a CPI of 1.8 for some program, and Computer B has a clock cycle time of 300 ps and a CPI of 1.25 for the same program. Which computer is faster for this program and by how much? Show your work.

**Time A = IC \* CPI \* CT = 200 \* 1.8 = 360ns \* IC**

**Time B = 300 \* 1.25 = 375ns \* IC**

**Computer A is faster by a rate of 375/360**

1. **(3 pts)** The Icestorm Company is rewriting its best-selling game, World of Minecraft, to better utilize the multicore hardware. If the system is using an 8-core processor, the revision allows 60% of the computation be perfectly parallelized with speedup by a factor of 4, and two thirds (2/3 or say 67%) of the parallelized computation can be further speed up by a factor of 8 in a computer with an 8-core processor.

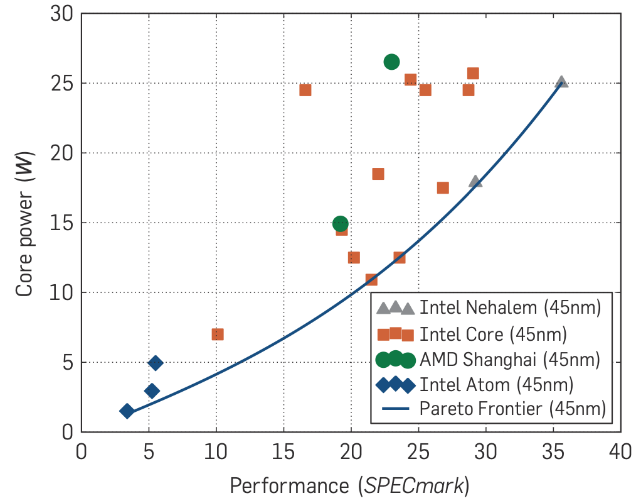
What is the total speedup of the revised World of Minecraft on an 8-core processor?

**Let speed on single core be X.**

**Then 8-core would be .4X/8 + .2X/4 + .4X = .8X/4 + .4X = .6X**

**So speedup would be Time baseline/ Time multicore = X/.6x = 1/.6 = 1.666x speedup**

1. **(3 pts)** In the following figure, what is the blue line called? Describe the concept that the name associated with the line

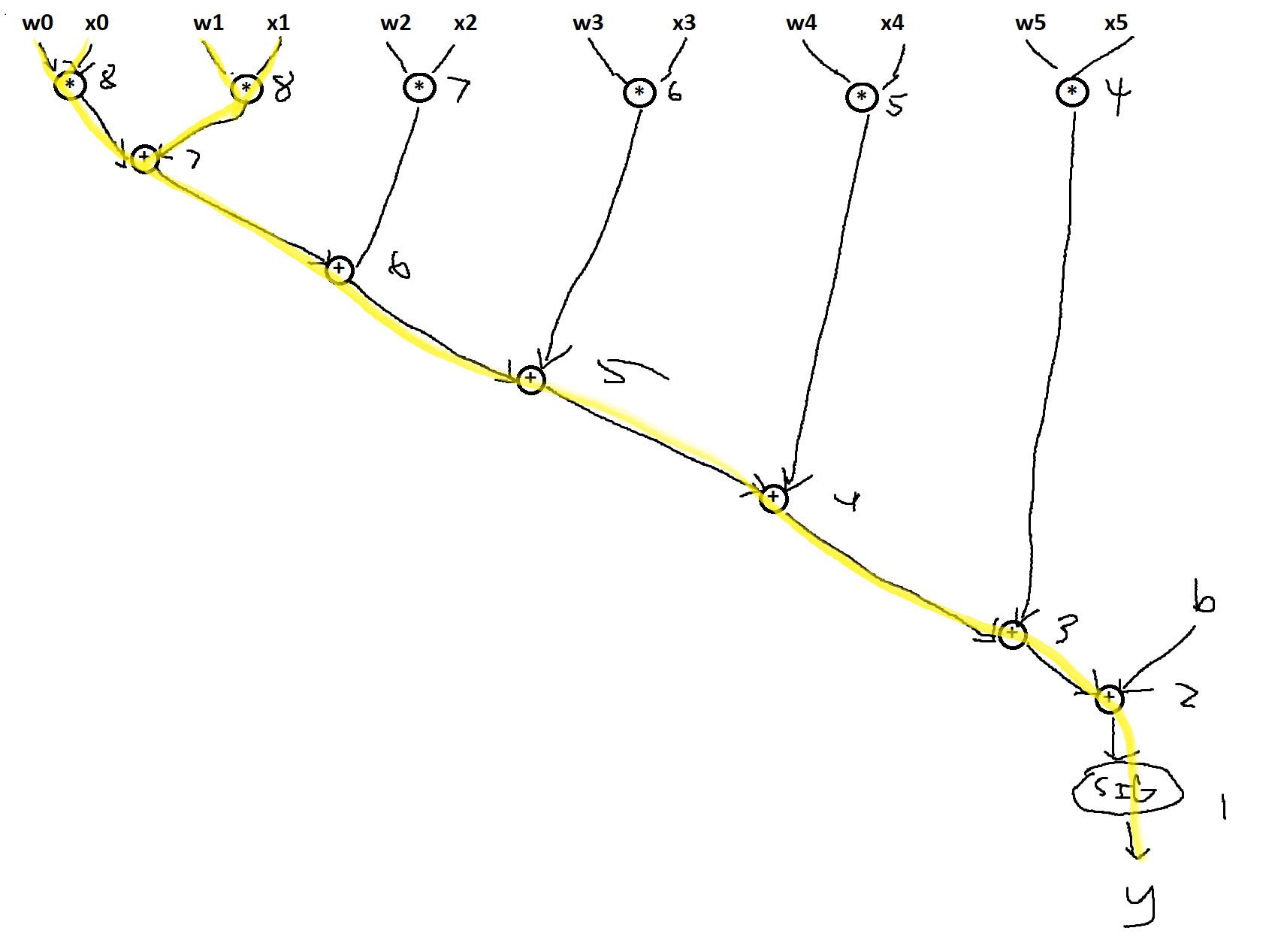
. 

**It is called the Pareto Optimal Frontier, the set of all Pareto efficient processors, which annotates diminishing returns on different processors not in the set**

1. **(11 pts) Dataflow Graphs**

Consider the following mathematical expression:

* 1. **(2 pt)** First, draw the DFG for this expression.



* 1. **(3 pts)** Calculate the number of edges that takes from each node of the graph to reach y and write it by the corresponding node in your drawing. This number is called distance from the output.
  2. **(2 pt)** Highlight the longest path(s) from any input to the output that passes through the maximum number of nodes. (You can use a different color for highlighting the pass.)
  3. **(4 pts)** If you are given two ALUs that support add, mult, and sigmoid operation, what is the feast number of cycles that you can execute the above expression. Each operation takes one cycle.

**There are 8 operations we need to perform, which we can complete in 4 cycles with two ALUs**

1. **(15 pts) Instruction Set Architecture (ISA)**
2. **(9 pts)** Consider a processor with an instruction length of 16 bits and 16 general purpose registers. Each register holds 16 bits. For each of the following scenarios, explain whether it is possible to have the described instruction set.
   1. **(3 pts)** 10 instructions that use 2 register operands and 1 shamt operand

**We can fit 10 instructions with 4 bits for opcodes, and 8 bits for two registers. Since shamt should be able to shift 0-16 values, 17 total, we would need 5 bits to encode it, 4 + 8 + 5 = 17 > 16 so it is not possible**

* 1. **(3 pts)** 50 instructions that use 3 register operands

**We would need at least 6 bits to represent 50 distinct opcodes, and 12 for 3 registers. 6 + 12 = 18 > 16 so it would not be possible**

* 1. **(3 pts)** 33 instructions that use 1 register operand and an immediate operand that needs to be able to represent all unsigned values between 0 and 50

**We would need 6 bits for opcodes for the instructions, 4 for the register. For the unsigned values, we would need 101 values to be represented, which we can with 7 bits. 6 + 4 + 7 = 17 > 16 so it would not be possible**

1. **(6 pts)** Write comparable C code for the following MIPS instructions.

# $s0 is assumed to hold a base address of an integer array  
function\_dsf:   
**1** multi $a0, $a0, 4  
**2** add $t0, $s0, $a0  
**3** lw $t1, 0($t0)   
**4** multi $t1, $t1, 4  
**5** beq $t1, $a0, end  
**6** loop: add $t0, $s0, $t1   
**7** add $t2, $t1, $zero  
**8** lw $t1, 0($t0)  
**9** multi $t1, $t1, 4  
**10** bne $t1, $t2, loop   
**11** end: divi $t1, $t1, 4  
**12** add $v0, $t1, $zero  
**13** jr $ra

multi is a multiply instruction with two operands: register and immediate.  
divi is a divide instruction with two operands: register and immediate.

Dsf(&A, &s[], &Vo) {

A \*= 4;

Int Tz = s[A];

To = Tz;

To \*= 4;

If (To != A) {

Int Tt;

While (To != Tt) {

Tz = s[To];

Tt = To + 0;

To = Tz;

To \*= 4;

}

}

To /= 4;

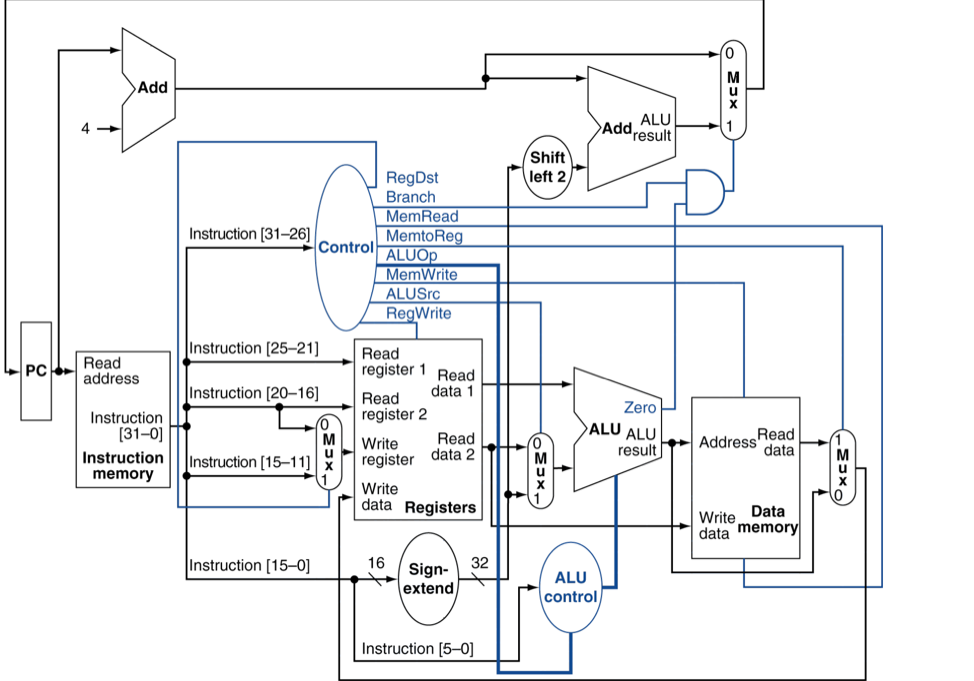
Vo = To + 0;

return;

}

1. **(6 pts) Single-cycle Processor**

# For the following single-cycle data path:



# **(3 pts)** Consider a beq instruction. Enter 1, 0, or x (for don’t care) in the table below for the various outputs of the control unit for this instruction.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RegWrite** | **RegDst** | **Branch** | **MemRead** | **MemtoReg** | **MemWrite** | **ALUSrc** |
| **0** | **X** | **1** | **0** | **X** | **0** | **0** |

# **(3 pts)** Consider an R-type instruction considering that the index of the destination register is in bits 15:11 of the instruction. Enter 1, 0, or x (for don’t care) in the table below for the various outputs of the control unit for this instruction.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **RegWrite** | **RegDst** | **Branch** | **MemRead** | **MemtoReg** | **MemWrite** | **ALUSrc** |
| **1** | **1** | **0** | **0** | **0** | **0** | **0** |

1. **(15pts) Single-cycle/Multi-cycle Processor**

Given these timings for individual stages of the datapath:

IF: 200ps, ID: 70ps, EX: 190ps, MEM: 300ps, WB: 140ps.

Assume the processor is designed to execute all the usual instructions unless stated otherwise.

1. **(3 pts)** What is the clock cycle time and frequency of a single-cycle processor? Also, what is this processor throughput assuming that instructions are equally distributed among store, load, and r-type instructions?

**CT = 200 + 70 + 190 + 300 + 140 = 900ps**

**Latency = 900ps**

**Frequency = 1/900ps**

**Throughput = 1 inst / 1 cycle \* 1 cycle / 900ps**

1. **(3 pts)** How long does it take for an add, store, and load instruction to execute on a single-cycle processor?

**All instructions take one cycle, so they all take to CT which is 900ps**

1. **(3 pts)** What is the clock cycle time and frequency of the multi-cycle processor? Also, what is the throughput assuming that instructions are equally distributed among store, load, and r-type instructions?

**CT = Max of stages = 300ps**

**Frequency = 1/300ps**

**Throughput = 1 inst / 1 cycle \* 1 cycle / 300ps**

1. **(3 pts)** How long does it take for an add, store, and load instruction to execute on a multi-cycle processor?

**Add is R-type, which takes 4 cycles, so 4 \* 300 = 1200ps**

**Ld is 5 \* 300 = 1500ps**

**Str is 1500ps**

1. **(3 pts)** For both single-cycle and multi-cycle processors, if you could split one of the stages into 2 equal halves, which one would you choose? What is the new cycle time, new latency, and new throughput?

**Choose MEM as it has the longest latency. Single-cycle Cycle Time, latency, and throughput doesn’t change.**

**New:**

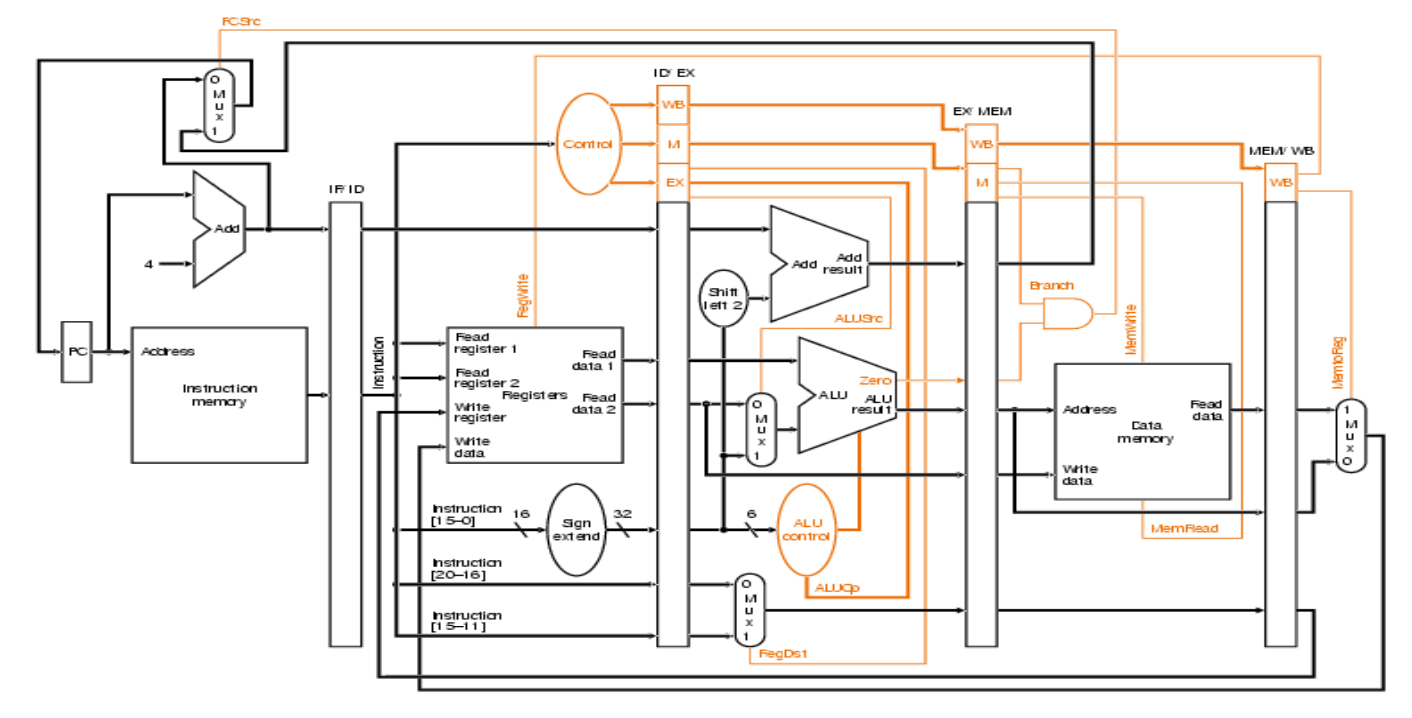
**CT = 200 + 20 = 220ps**

**Latency = 220 \* 6 = 1320ps**

**Throughput = 1 inst / 1 cycle \* 1 cycle / 220ps**

1. **(6 pts) Pipelined Processor Design**

For the given MIPS pipeline, consider a stall that occurs in the decode stage (meaning that a hardware *noop* is pushed into the EX stage). To clarify, this means the instructions in the IF and ID stages must remain the same.

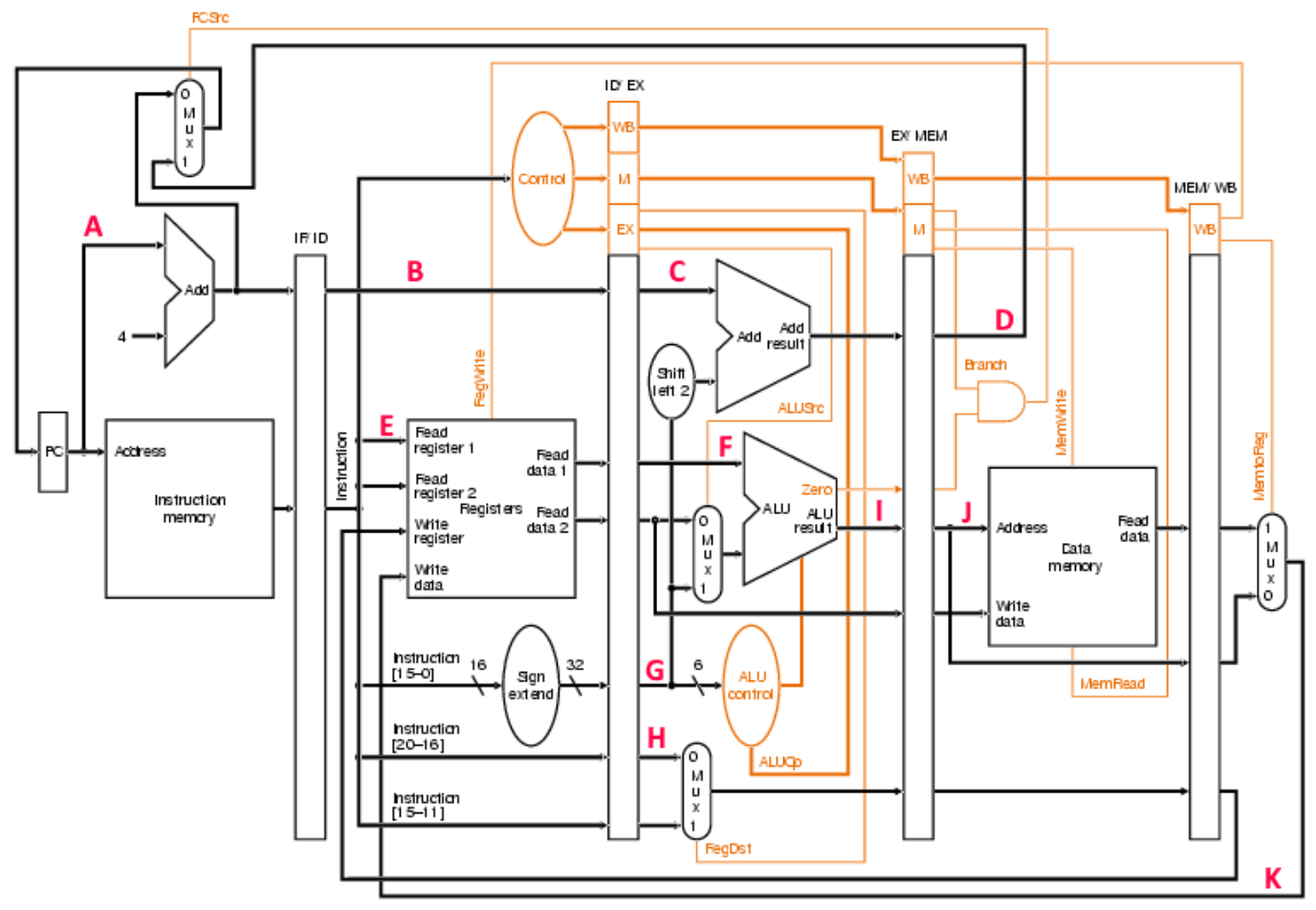


1. **(3 pts)** For the inserted hardware noop instruction, list the critical control signals being written to the ID/EX register and their values. A control signal is considered critical if the signal cannot be don't-care without incorrectly setting or corrupting a register, PC, or memory value.
2. **(3 pts)** What additional control signals would be needed to successfully stall the pipeline? Explain why they are needed.
3. **(12 pts) Pipelined Processor Design**

The figure below shows the pipelined datapath with the control signals connected to the control portions of the pipeline registers. Forwarding is available wherever needed as discussed in the class. Consider the following set of instructions. (R0 register is wired to the value “0”)

**1** LW R1, 20(R0)   
 **2** LW R1, 0(R1)  
 **3** ADDI R2, R0, 0  
 **4** SW R2, 0(R1)  
 **5** ADDI R2, R2, 4  
 **6** SW R2, 4(R1)  
 **7** ADDI R2, R2, 4  
 **8** SW R2, 8(R1)

* 1. **(6 pts)** For the given pipeline diagram, provide the *value of the datapath* wires marked with red letters in *cycle 7*. Also state whether the value at that wire is garbage (a value is garbage if it will not be used to change any MIPS register, memory, or the PC value). Assume that if the IF stage of the first instruction (LW) executes in cycle 1. The value in data memory is the same as the address plus 100 (e.g., address 60 holds the value 160). Also assume that the first instruction has a PC value of 0x0000. Write your answers in the provided table.



|  |  |  |
| --- | --- | --- |
| Instruction | Value | Is Garbage? |
| A |  |  |
| B |  |  |
| C |  |  |
| D |  |  |
| E |  |  |
| F |  |  |
| G |  |  |
| H |  |  |
| I |  |  |
| J |  |  |
| K |  |  |

* 1. **(6 pts)** What are the values of the following control signals in cycle 7? You must state if a signal is 0, 1, or don’t care.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| PCSrc | Branch | RegWrite | RegDst | ALUSrc | ALUOp | MemWrite | MemToReg |
|  |  |  |  |  |  |  |  |

1. **(15 pts) Data Hazards**
   1. **(3 pts)** Consider following assembly-language program:  
        
      **1** MOV R3, R7   
      **2**  LD R8, (R3)  
      **3** label: ADD R3, R3, 4  
      **4** LD R9, (R3)  
      **5**  BNE R8, R9, label  
        
      This program includes WAW, RAW, and WAR dependencies. Show all the dependences and qualify their kind. If instruction #i is dependent on instruction #j, write it as #i -> #j. For two or more dependencies, separate them with a comma (#i -> #j, #m -> #n).

**WAW (1 -> 3, 2 -> 5)**

**RAW (1 -> 3, 1 -> 4, 4 -> 5, 3 -> 4)**

**WAR (2 -> 3)**

* 1. **(6 pts)** Assume a 5-stage pipelined processor (IF, ID, EX, MEM, WB) with forwarding as discussed in the class. Show how the instructions below would progress though this 5-stage pipeline.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ADD R5, R2, R1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW R3, 4(R5) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW R2, 0(R2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| OR R3, R5, R3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SW R3, 0(R5) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

# 

# **(6 pts)** Assume you break up the memory stage into 2 stages instead of 1 to improve throughput in a pipelined datapath. Thus, the pipeline stages are now: IF, ID, EX, MEM1, MEM2, WB. Show how the instructions below would progress though this six-stage pipeline. Full forwarding hardware is available.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| ADD R5, R2, R1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW R3, 4(R5) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| LW R2, 0(R2) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| OR R3, R5, R3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| SW R3, 0(R5) |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

1. **(6 pts) Branch Prediction**

Compute the branch prediction accuracy for the following two branch traces with the 1-bit and 2-bit branch predictors. The 1-bit predictor starts in the ’Not-Taken’ and the 2-bit predictor starts in the ’Weakly-Taken’ state at the beginning of each branch trace. Provide your solution inline below, as well as your work. You will not be given full credit without sufficient description.

1. **(3 pts)** Trace: T N T N T T N N T T N N T

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T | N | T | N | T | T | N | N | T | T | N | N | T |
| 1-bit | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| Hit/miss | X | X | X | X | X | O | X | O | X | O | X | O | X |
| 2-bit | 10 | 11 | 10 | 11 | 10 | 11 | 11 | 10 | 10 | 11 | 11 | 10 | 10 |
| Hit/miss | O | X | O | X | O | O | X | X | O | O | X | X | O |

1-bit accuracy is: 4/13 = 30.8%  
2-bit accuracy is: 7/13 = 53.8%%

1. **(3 pts)** Trace: T T N T T N T N T N T N T

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | T | T | N | T | T | N | T | N | T | N | T | N | T |
| 1-bit | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| Hit/miss | X | O | X | X | O | X | X | X | X | X | X | X | X |
| 2-bit | 10 | 11 | 11 | 10 | 11 | 11 | 10 | 11 | 10 | 11 | 10 | 11 | 10 |
| Hit/miss | O | O | X | O | O | X | O | X | O | X | O | X | O |

1-bit accuracy is: 2/13 = 15.4%  
2-bit accuracy is: 8/13 = 61.5%

1. **(12 pts) Branch Prediction**

Given the following snippet of code below, compute the accuracies of the 1-bit and 2-bit predictors (fill the table below). Assume that the branch is taken if the “if” condition is met. The 1-bit predictor is initialized to 1 and the 2-bit predictor is initialized to 10. Show your work.

i = 0;  
do {  
 if (i % 2 == 0) { **# Branch 1**  
 …  
 }  
 if (i % 2 == 1) { **# Branch 2**  
 …  
 }  
 ++i;  
} while (i < 10000); **# Branch 3**

**i: 0, 1, 2, 3… 9999**

**Branch1 Actual: T N T N... N**

**1 Bit Thus only predicts first one accurately, accuracy = 1/10000 = .01%**

**2 Bit predicts all the T correctly, so it is 50%**

**Branch2 Actual: NT T NT T… T**

**1 Bit Thus predicts all wrong, accuracy = 0%, 1 0 1 0 1 0**

**2 Bit also predicts all wrong, as it will predict 10, 01, 10…**

**Branch3 Actual: 1 1 … 1, will predict last one wrong for 1 bit and 2 bit, as when i is 10000 it will guess T**

**Accuracy = 9999/10000 = 99.99%**

|  |  |  |
| --- | --- | --- |
|  | **1-bit** | **2-bit** |
| **Branch 1** | **.01%** | **50%** |
| **Branch 2** | **0%** | **0%** |
| **Branch 3** | **99.99%** | **99.99%** |

1. **(8 pts) Cache**

# A processor has a separate D-cache and an I-cache.

# D-Cache: 128KB, 8-way set associative, block size of 1 word I-Cache: 64KB, direct mapped cache, block size of 2 words

# Answer the following questions. The processor uses 32 bits for its address.

# **(2 pts)** Calculate the number of tag, index, and offset bits for the D-cache.

# **1 word = 4 bytes**

# **Offset = log(4) = 2 bits**

# **128 KB = 2^7 \* 2^10 = 2^17 = 131072 bytes**

# **128KB/4 = 32768 bytes**

# **32768 / 8 = 4096 = 2^12, Index bits = 12. Tag bits = 32 – 12 – 2 = 18.**

# **Tag: 18, Index: 12, Offset: 2**

# **(2 pts)** Calculate the number of tag, index, and offset bits for the I-cache.

# **2 words = 8 bytes**

# **Offset = log(8) = 3 bits**

# **64 KB = 2^6 \* 2^10 = 2^16 = 65536 bytes**

# **64KB/8 = 8192 Bytes = 2^13, Index bits = 13. Tag bits = 32 – 13 – 3 = 16**

# **Tag: 16, Index: 13, Offset: 3**

# **(2 pts)** How many bits are needed to implement the D-cache?

# **Default -> tag = 18 bits, valid bit, data = 32 bits**

# **Total = 18 + 1 + 32 = 51 bits per block, so 2^15 \* 51 bits**

1. **(2 pts)** How many bits are needed to implement the I-cache?

**Default -> tag = 16 bits, valid bit, data = 32 bits**

49 bits per block, so **2^13 \* 49 bits**