# EECS 470 *Midterm Exam*

*Winter 2025*

Name: _____ Uniqname: _____

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

---

| # | Points |
|---|---|
| 1 | **/ 10** |
| 2 | **/ 16** |
| 3 | **/ 16** |
| 4 | **/ 17** |
| 5 | **/ 17** |
| 6 | **/ 10** |
| 7 | **/14** |
| ***Total*** | **/100** |

# NOTES:

- Closed book and one 8.5x11 page of notes
- Calculators are allowed, but no smart watches, cell phones, earbuds, etc.
- Don't spend too much time on any one problem.
- You have about 120 minutes for this exam.
- There are **13** pages including this one.
- **Be sure to show your work and explain what you've done when asked to do so.**

**1) Short Answer - Fill in the correct bubbles [10 Points]:**

a) In R10k, with a ROB size of 32, RS size of 16, and ARF size of 32, what is the maximum number of PRF entries you would expect to have?

◯ **32** ◯ **36** ◯ **48** ◯ **64** ◯ **80**

b) If you have a pipelined multiplier on the critical path of your processor, what would you expect to happen if you decrease the number of pipeline stages?

◯ **CPI up, clock freq up**
◯ **CPI up, clock freq down**
◯ **CPI down, clock freq up**
◯ **CPI down, clock freq down**

c) Adding more reservation station entries will reduce the amount of time that the processor stalls due to **(i)** ***structural hazards / data hazards / control hazards*** but may increase the **(ii)** ***PRF size / clock period / clock frequency / CPI***.

Choice (i):
◯ **structural hazards**
◯ **data hazards**
◯ **control hazards**

Choice (ii):
◯ **PRF size**
◯ **clock period**
◯ **clock frequency**
◯ **CPI**

d) In an R10k processor, which of the following structures are modified following an exception? **Select all that apply**.

◯ **ROB** ◯ **RS** ◯ **Map Table** ◯ **Arch. Map Table**

e) Reducing supply voltage may generally result in a **(i)** ***linear / quadratic / cubic*** improvement to power consumption and a (ii) ***linear / quadratic / cubic*** degradation in performance.

Choice for (i):
◯ **linear**
◯ **quadratic**
◯ **cubic**

Choice for (ii):
◯ **linear**
◯ **quadratic**
◯ **cubic**

f) The ***avoidance / detect and stall / detect and forward / speculate and squash*** method for data hazard avoidance involves inserting NOP instructions into the program using the compiler.

◯ **avoidance** ◯ **detect & stall** ◯ **detect & forward** ◯ **speculate & squash**

**2) Scoreboarding vs Tomasulo's [16 Points]:**

Look at the snapshots of programs below running on a 1-way machine with 5 functional units: 1 ALU, 1 Load, 1 Store, and 2 FP (both can handle all types of arithmetic FP instructions and have a 3 cycle latency). Identify what type of hazard(s) (if any) will occur on the next cycle if the machine is:

a) The Scoreboarding scheduling algorithm as presented in lecture
b) Tomasulo's scheduling algorithm as presented in lecture

Note that f0, f1, f2 are floating point registers and r0, r1, r2 are standard registers.

| Insn Status | | | | |
|---|---|---|---|---|
| Inst | D | S | X | W |
| 1) r2 = ld([r1] + 8) | c1 | c2 | c3 | |
| 2) f0 = f2 + f0 | c2 | c3 | | |
| 3) r1 = r2 + r2 | c3 | | | |
| 4) r0 = r1 + r1 | | | | |

Scoreboarding:

Tomasulo's:

---

| Insn Status | | | | |
|---|---|---|---|---|
| Inst | D | S | X | W |
| 1) f0 = ldf([r1] + 8) | c1 | c2 | c3 | |
| 2) r1 = r2 + r2 | c2 | c3 | | |
| 3) f0 = f1 + f2 | | | | |
| 4) f2 = f1 * f1 | | | | |

Scoreboarding:

Tomasulo's:

---

| Insn Status | | | | |
|---|---|---|---|---|
| Inst | D | S | X | W |
| 1) f0 = ldf([r1] + 8) | c1 | c2 | c3 | c4 |
| 2) f1 = f0 * f0 | c2 | c4 | c5+ | |
| 3) ([r1] + 24) = stf(f1) | c3 | | | |
| 4) r1 = r2 + r0 | c4 | c5 | c6 | |
| 5) f2 = f0 + f1 | c5 | | | |

Scoreboarding:

Tomasulo's:

**3) Power and Performance [16 Points]:**

a) Bits'N'Bobs, a local chip manufacturer, is working on producing more efficient chips. Their baseline processor runs at 80W, 1 BIPS (billion instructions per second), and they are considering implementing one of the following new operating modes. Which of the options below, if any, provide an improvement if implemented? Assume that at most one mode will be added, so only compare with the baseline. **[4 Points]**

- Disable 30% of the on-chip caches: 65W, 900 MIPS
- Decrease ROB size by 20%: 55W, 750 MIPS

b) The marketing execs at Somewhat Satisfactory Software Solutions Inc, another local chip manufacturer, are looking for performance averages to use in promotional material for their newest flagship processor.

i) They first run a series of benchmarks that provide the following Instructions Per Cycle (IPC) values: 2.40, 1.63, 1.54, 1.96. What averaging method would you use here? What is the processor's average IPC? **[3 Points]**

ii) They then compared their processor's benchmark times to those of their top three competitors and found the following speedups: 1.33x, 0.98x, and 1.22x better, respectively. What averaging method would you use here? What is the processor's average speedup compared to others on the market? **[3 Points]**

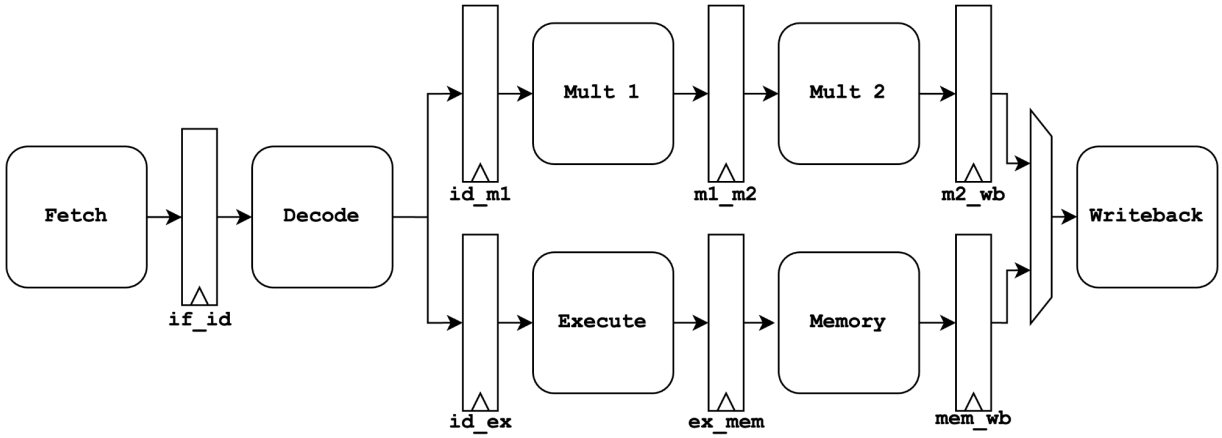
c) Consider a program consisting of two functions (func1 and func2). When executed serially on a single core, the program takes 1 second to run, with 20% of execution time spent in func1 and 80% in func2. Func2 can be executed in parallel across an arbitrary number of cores with perfectly linear speedup (i.e. executing across n cores means func2 runs n times as fast).

i) Calculate the execution time when the program is **parallelized across 8 CPU** cores. **[3 Points]**

ii) Func2 can be offloaded to a data-parallel accelerator with 80 cores. However, each of these cores is half as fast as the CPU cores. It takes *x* amount of time to transfer func2 to the accelerator and get data back. **For what values of *x* does offloading to the accelerator yield a performance benefit compared to the 8 core CPU**. Answer as an inequality. **[3 Points]**

**4) In Order Pipeline [17 Points]:**

One big flaw in the project 3 pipeline is that the only way to perform multiplication is to execute repeated additions. With 32-bit numbers and a 10ns clock period, the worst-case multiplication will take $2^{32}$ cycles or over 4 seconds! To fix this, you are asked to add a 2-stage multiplier. Stage 1 is placed with the EX stage and stage 2 occurs with the MEM stage, so the pipeline now looks as follows:




a) What new forwarding paths does this modification introduce? Check the corresponding boxes to indicate the forwarding source and forwarding destination. **Include all valid forwarding paths, even if we didn't use them in project 3**. You may or may not need to use all the rows in the table. **[10 Points]**

| | Data Forwarding Source | | | | Data Forwarding Destination | | | |
|---|---|---|---|---|---|---|---|---|
| | ex_mem | mem_wb | m1_m2 | m2_wb | Execute | Memory | Mult 1 | Mult 2 |
| Path 1 | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Path 2 | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Path 3 | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Path 4 | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |
| Path 5 | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ | ◯ |

b) What new data hazard(s) do we now have to check for? If none, justify why not. **[3 Points]**

c) When benchmarking this pipeline, you discover that most of the multiplies have operands that are only of 8-bits or less. With numbers this small, they only require one multiply stage to execute. What optimization could you add to the microarchitecture to speed up program execution based on this insight? **[4 Points]**

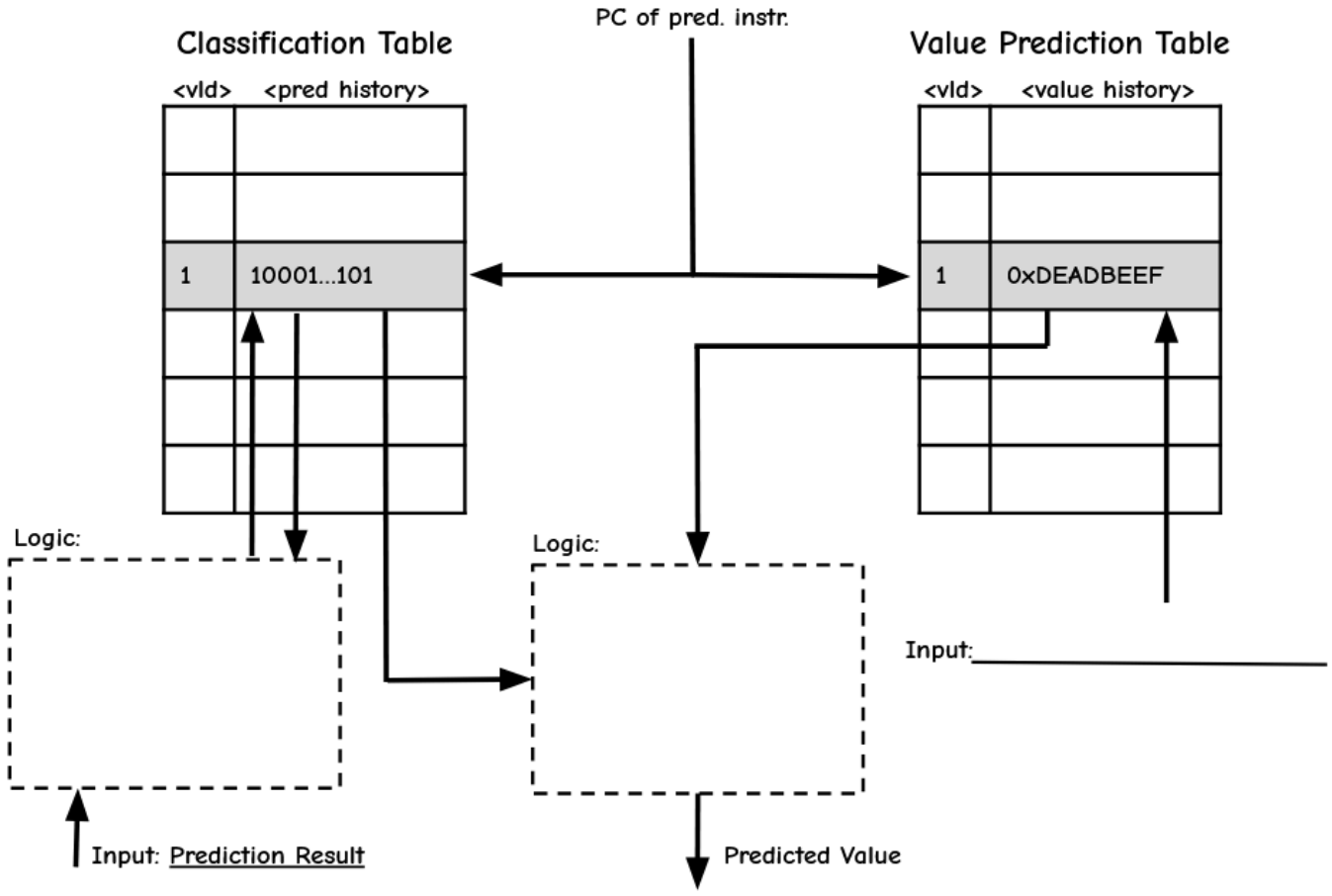
**5) Value Prediction [17 Points]**

In 1996, Lipasti and Shen (Carnegie Mellon) proposed **Value Prediction** as a solution to the serialization imposed by true data dependencies (read-after-writes). The authors observed that programs exhibit **value locality** – the tendency for the same value to reappear in a given storage location (register). If we can accurately predict register values based on history, then we can execute dataflow-dependent instructions in parallel!

a) In order to predict values, the authors implement the **Value Prediction Unit (VPU)**. The VPU consists of two tables, both indexed using the PC of the instruction we are predicting for.

1. The **classification table** keeps a record of the prediction history (whether or not a given instruction had its value predicted correctly) via a saturating counter
2. The **value prediction table** holds the last correct value for a given instruction

In this problem, we will output the predicted value if the relevant classification table counter exceeds a threshold value. Otherwise, we will output a default value. Assume a threshold value of 3 and a default value of 0. Complete the figure below by drawing the logic necessary to **update the classification table** using the prediction result and **produce a prediction** using the outputs of each table. Also identify the input needed to **update the value prediction table**. **[8 Points]**



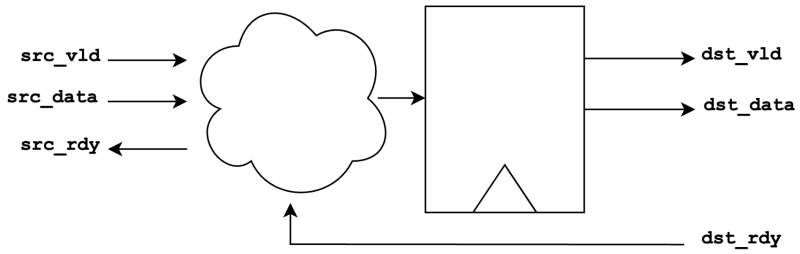
b) Because indexing the tables using the entire PC would require prohibitively large tables (2^32 entries for a 32 bit PC), we might choose to use a subset of the PC bits. For a very area-constrained processor, say we select bits 2-6 of the PC as our index bits. Briefly explain how this choice could cause a degradation in our performance. **[3 Points]**

c) Because of value prediction, instructions which consume a register may now begin executing even while that register's producer has not yet completed. However, when the consumer instruction issues, it may not immediately yield its reservation station entry. Explain why this is necessary to maintain correctness. **[3 Points]**

d) Briefly explain **two** potential sources of lost cycles presented by our implementation of value prediction. **[3 Points]**

**6) SystemVerilog Design [10 Points]:**
Backpressure is a design technique in which one pipeline stage stalls if the stage following it cannot accept any new data. To implement this, many designers choose to use a valid-ready interface in which the source pipestage asserts a “valid” signal when it has data to send to the next stage, and the consuming stage asserts a “ready” signal when it can accept new data. Data only moves from one pipestage to the next when both ready and valid are high, and this ensures that no data is lost between stages.

To modularize this approach, you are asked to design a parameterized module that implements a pipeline stage between two modules using the ready-valid handshake. Complete the module implementation according to the diagram below.




```
module VldRdySlice #(parameter WIDTH = 32)
   (input                    clock, reset,
    input                    src_vld,
    output logic             src_rdy,
    input        [WIDTH-1:0] src_data,
    output logic             dst_vld,
    input                    dst_rdy,
    output logic [WIDTH-1:0] dst_data);
```

```
Endmodule
```

**7) R10K and P6 Walkthrough [14 Points]**

On the next pages, you will find a set of charts showing a snapshot of a MIPS R10k-like microarchitecture and a P6-like microarchitecture after five cycles executing a sequence of instructions. You must advance this machine 1 additional clock cycle (to the end of cycle #6). Use the state tables to record the contents of each hardware structure at the end of the sixth clock cycle.

Assume the following:

- Assume the machine is a 2-wide superscalar (e.g., it can issue, complete, and retire at most 2 instructions per cycle). If there are conflicts among instructions, the machine always selects the oldest instructions first.
- Ignore the fetch stage. Assume all instructions have been fetched and are ready for dispatch whenever the out-of-order core allows.
- This machine has 4 architectural registers, a 5-entry ROB, 4 reservation stations, and 8 physical registers (for the R10k).
- There are **2 add** functional units with a 1-cycle latency, and **1 fully-pipelined multiply** functional unit with a **2-cycle** latency (fully-pipelined means the multiply unit has 2 pipeline stages; it can issue a new multiply each cycle, however, multiplies take 2 cycles to execute).
- Assume there is no bypassing in the X stage, but C will bypass to S through the physical register file. Assume reservation stations are freed at the same place as in lecture, and can be reused as soon as they are freed.
- This is the code that is being executed:
    - (1) R1=R1*R3
    - (2) R4=R4*10
    - (3) R1=R3+R2
    - (4) R2=R2+5
    - (5) R1=R1+R3
    - (6) R1=R1*R3
    - (7) R4=R4*10
    - (8) R1=R3+R2

Be sure to update the head and tail pointers correctly to show when instructions are dispatched and retired.

a) Complete Cycle 6 for the R10K-style Machine **[7 Points]**

# R10K Cycle # 5

| ROB | | | | | | | |
|---|---|---|---|---|---|---|---|
| ht | # | Insn | T | Told | S | X | C |
| h | 1 | I1: R1 = R1 * R3 | p8 | p1 | c2 | c3+ | c5 |
| | 2 | I2: R4 = R4 * 10 | p4 | p5 | c3 | c4+ | |
| | 3 | I3: R1 = R3 + R2 | p7 | p8 | c5 | | |
| t | 4 | I4: R2 = R2 + 5 | p3 | p2 | c3 | c4 | c5 |
| | 5 | I5: R1 = R1 + R3 | | | | | |

| Map Table | |
|---|---|
| Reg | T+ |
| r1 | p7 |
| r2 | P3+ |
| r3 | P6+ |
| r4 | p4 |

| Arch Map | |
|---|---|
| Reg | T |
| r1 | p1 |
| r2 | p2 |
| r3 | p6 |
| r4 | p5 |

| Reservation Stations | | | | | |
|---|---|---|---|---|---|
| # | busy | op | T | T1 | T2 |
| 1 | y | I3 | p7 | p6+ | p2+ |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |

| CDB |
|---|
| T |
| p8, p3 |

| Free List |
|---|
| |

# R10K Cycle # 6

| ROB | | | | | | | |
|---|---|---|---|---|---|---|---|
| ht | # | Insn | T | Told | S | X | C |
| | 1 | | | | | | |
| | 2 | | | | | | |
| | 3 | | | | | | |
| | 4 | | | | | | |
| | 5 | | | | | | |

| Map Table | |
|---|---|
| Reg | T+ |
| r1 | |
| r2 | |
| r3 | |
| r4 | |

| Arch Map | |
|---|---|
| Reg | T |
| r1 | |
| r2 | |
| r3 | |
| r4 | |

| Reservation Stations | | | | | |
|---|---|---|---|---|---|
| # | busy | op | T | T1 | T2 |
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |

| CDB |
|---|
| T |
| |

| Free List |
|---|
| |

b) Complete Cycle 6 for the P6-style machine **[7 Points]**

# P6 Cycle #5



| ROB | | | | | | | |
|---|---|---|---|---|---|---|---|
| ht | # | Insn | R | V | S | X | C |
| h | 1 | I1:R1 = R1 * R3 | R1 | CDB.V1 | c2 | c3+ | c5 |
| | 2 | I2:R4 = R4 * 10 | R4 | | c3 | c4+ | |
| | 3 | I3:R1 = R3 + R2 | R1 | | c5 | | |
| | 4 | I4:R2 = R2 + 5 | R2 | CDB.V2 | c3 | c4 | c5 |
| t | 5 | I5:R1 = R1 + R3 | R1 | | | | |

| Map Table | |
|---|---|
| Reg | Tag |
| r1 | ROB#5 |
| r2 | ROB#4+ |
| r3 | |
| r4 | ROB#2 |

| Reservation Stations | | | | | | |
|---|---|---|---|---|---|---|
| # | Insn# | T | T1 | T2 | V1 | V2 |
| 1 | I5 | ROB#5 | ROB#3 | - | - | [R3] |
| 2 | | | | | | |
| 3 | I3 | ROB#3 | - | - | [R3] | CDB.V2 |
| 4 | | | | | | |

| CDB | |
|---|---|
| T | V |
| ROB#1 | [R1] |
| ROB#4 | [R2] |

# P6 Cycle #6



| ROB | | | | | | | |
|---|---|---|---|---|---|---|---|
| ht | # | Insn | R | V | S | X | C |
| | 1 | | | | | | |
| | 2 | | | | | | |
| | 3 | | | | | | |
| | 4 | | | | | | |
| | 5 | | | | | | |

| Map Table | |
|---|---|
| Reg | Tag |
| r1 | |
| r2 | |
| r3 | |
| r4 | |

| Reservation Stations | | | | | | |
|---|---|---|---|---|---|---|
| # | Insn# | T | T1 | T2 | V1 | V2 |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |

| CDB | |
|---|---|
| T | V |
| | |