



---

The University of Michigan - Department of EECS

EECS370 – Introduction to Computer Organization

Midterm Exam 2  
November 20, 2008

---

Name: \_\_\_\_\_ **SOLUTIONS** \_\_\_\_\_

University of Michigan unickname: \_\_\_\_\_  
(NOT your student ID number!)

Open book, open notes. No laptops, PDAs, cell phones, etc. (calculators are ok). This exam has **6** sets of questions, **12** pages, and **100** points. Questions vary in difficulty; so it is strongly recommended that you do not spend too much time on any one question. For questions where a box is provided, please give your final answer in the box.

Question	Points
1 – T/F and short questions	/17
2 – LC-2K performance	/15
3 – Multicycle datapath	/18
4 – Pipelined datapath	/20
5 – Short cache questions	/15
6 – Cache simulation	/15
Total	/100

The rules of the Honor Code of the University of Michigan - College of Engineering apply for this exam.

Honor code pledge: I have neither given nor received aid on this examination, nor have I concealed any violations of the Honor Code.

Signature: \_\_\_\_\_  
(Exams without a signed pledge will not be graded)

## 1. True/False and short questions [17 points]

Indicate whether the following statements are True or False:

a) [3 pts] In the presence of data hazards, branch prediction can be used to improve pipeline performance.

True

**False**

b) [3 pts] The CPI of a superscalar pipeline is always less than or equal to 1.

True

**False**

c) [3 pts] The LRU (least-recently used) replacement policy works because programs exhibit temporal locality.

**True**

False

d) [4 pts] Doubling the associativity of a cache without changing its overall capacity or block size will increase the tag size by one bit.

**True**

False

e) [4 pts] In the following statement, fill in the blanks:

When selecting cache associativity, computer architects try to manage the trade-off

between access-time and hit-rate.

## 2. LC-2K performance [15 points]

Consider a version of the 5-stage LC-2K8 pipelined datapath discussed in class. Assume that the datapath has the following properties:

- The branch prediction scheme is “**predict never taken**”.
- Branches are resolved at some stage, which is not necessarily MEM.
- The following instructions have a load hazard with a lw instruction that immediately precedes them:
  - 10% of add instructions,
  - 10% of nand,
  - 5% of sw, and
  - 15% of lw.
- The clock frequency is 300 MHz.
- The I-cache and the D-cache have a 100% hit rate.

Given is the following dynamic instruction breakdown for program X:

- 10% of instructions are add,
- 10% are nand,
- 15% are sw,
- 25% are lw, and
- 40% are beq.

Moreover, 55% of the beq instructions are not taken. The CPI of program X is 1.425.

a) [6 pts] What is the component of the CPI due to load-hazard stalls?  
[Show your work for credit]

**add stall + nand stall + sw stall + lw stall**

$$\mathbf{0.1*0.1*1 + 0.1*0.1*1 + 0.05*0.15*1 + 0.15*0.25*1 = 0.065}$$

b) [6 pts] How many pipeline stages are squashed when a branch is mispredicted?  
[Show your work for credit.]

$$1.425 = 1 + \text{load stalls} + \text{branch stalls}$$

$$1.425 = 1 + 0.065 + 0.45 * 0.4 * x$$

$$x = 2$$

c) [3pts] In what stage of the pipeline is the branch resolved?  
[Outline your reasoning for credit.]

**2 stages squashed means branches are resolved in stage  $2+1 = 3$  (Execute)**

### 3. Multicycle datapath [18 points]

Consider extending the multicycle LC-2K8 architecture by adding the instruction *tadd* (ternary add), which adds three registers together and stores the result into a fourth register:

`tadd regA regB regC destReg //destReg = regA + regB + regC`

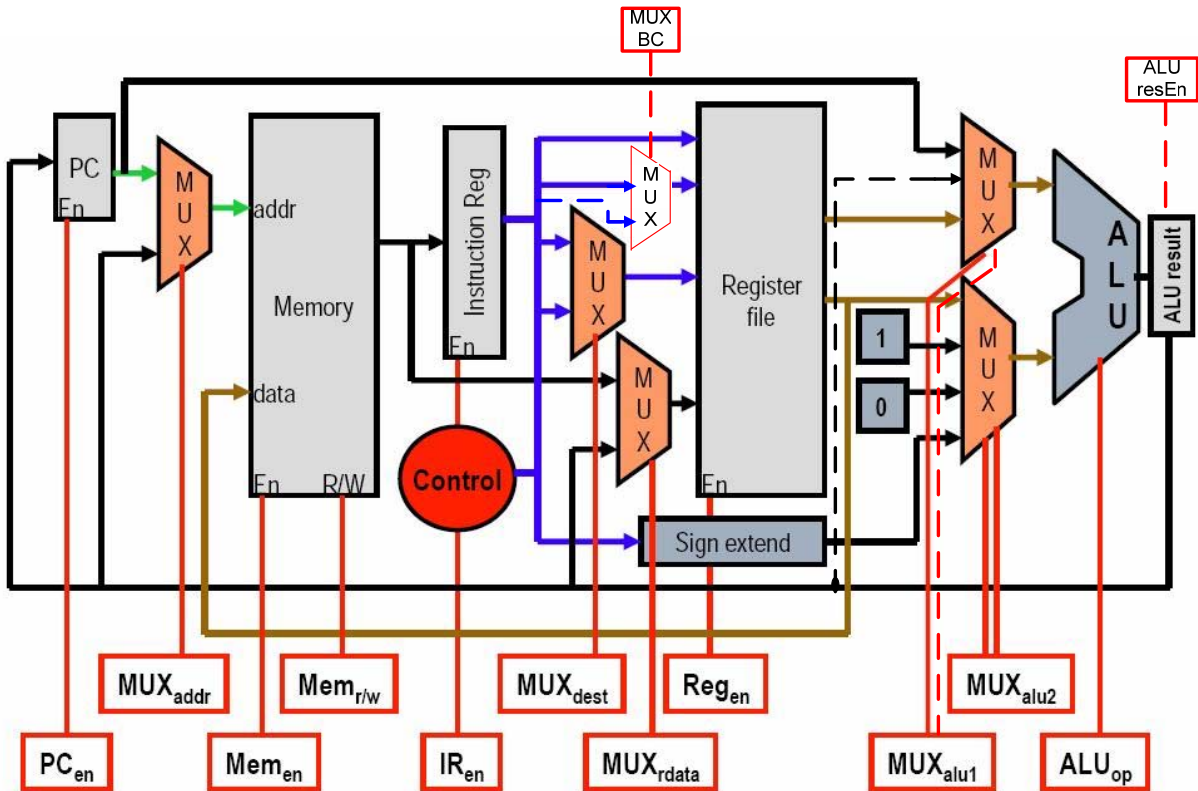
The instruction uses a new format, adapted from the R-type format (note that the opcode field is larger to accommodate the new instruction):

unused	opcode	regA	regB	regC	unused	destReg
31	26 25	22 21	19 18	16 15	13 12	3 2 0

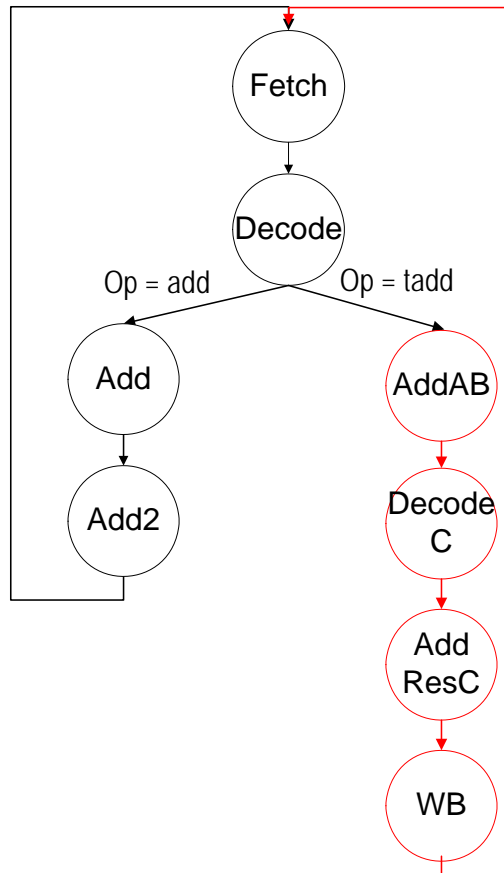
a) [6 pts] The multicycle datapath from lecture is provided below. Show what changes are needed to support *tadd*. You should only add wires, muxes, and new control signals to the datapath. **Do not modify the memory, register file, and ALU.** Assume that the datapath can already accommodate the larger opcode, and do not need to account for this in your modifications.

[Note: Full points will only be awarded to solutions that use a minimal number of cycles, do not lengthen the clock cycle, and use a minimal number of new components. Assume that the ALU, Memory and Register file all take 5ns, everything else is instantaneous.]

**Dotted lines are introduced.**



b) [4 pts] Add the states that are needed to implement *tadd* to the finite state machine below. The diagram already contains the states to implement the add instruction. You only need to draw the states for *tadd*, do not be concerned with other LC-2K8 instructions.



c) [8 pts] Fill in the following table, adding columns for any control signals you added in part a) above, and adding rows for any new state that you created in part b) above.

	PC en	MUX addr	Mem en	Mem r/w	IR en	MUX dest	MUX rdata	Reg en	MUX alu1	MUX alu2	ALU op	MUX regBC	ALU resEn		
Fetch	0	0	1	0	1	X	X	0	00	01	0	X	1		
Decode	1	X	0	X	0	X	X	0	X	X	X	0	1		
Add	0	X	0	X	0	X	X	0	10	00	0	0	1		
Add2	0	X	0	X	0	1	1	1	X	X	X	X	X		
Add A+B	0	X	0	X	0	X	X	0	10	00	0	0	1		
Decode C	0	X	0	X	0	X	X	0	X	X	X	1	0		
Add Result+C	0	X	0	X	0	X	X	0	01	00	0	1	1		
Write Back	0	X	0	X	0	1	1	1	X	X	X	X	X		

#### 4. Pipelined datapath [20 points]

As Chief Architect of SuperCausal, Inc., you are designing a high-performance version of the LC-2K8 pipelined datapath which eliminates load hazards with R-type instructions by introducing a forwarding path from the Mem/WB pipeline register to the Mem stage.

a) [7 pts] Show your new forwarding path in the figure on the next page. Make sure you show all wires and hardware components (multiplexers, adders, registers, etc.) that you introduce. Also, make sure you indicate all wires that are disconnected or re-routed.

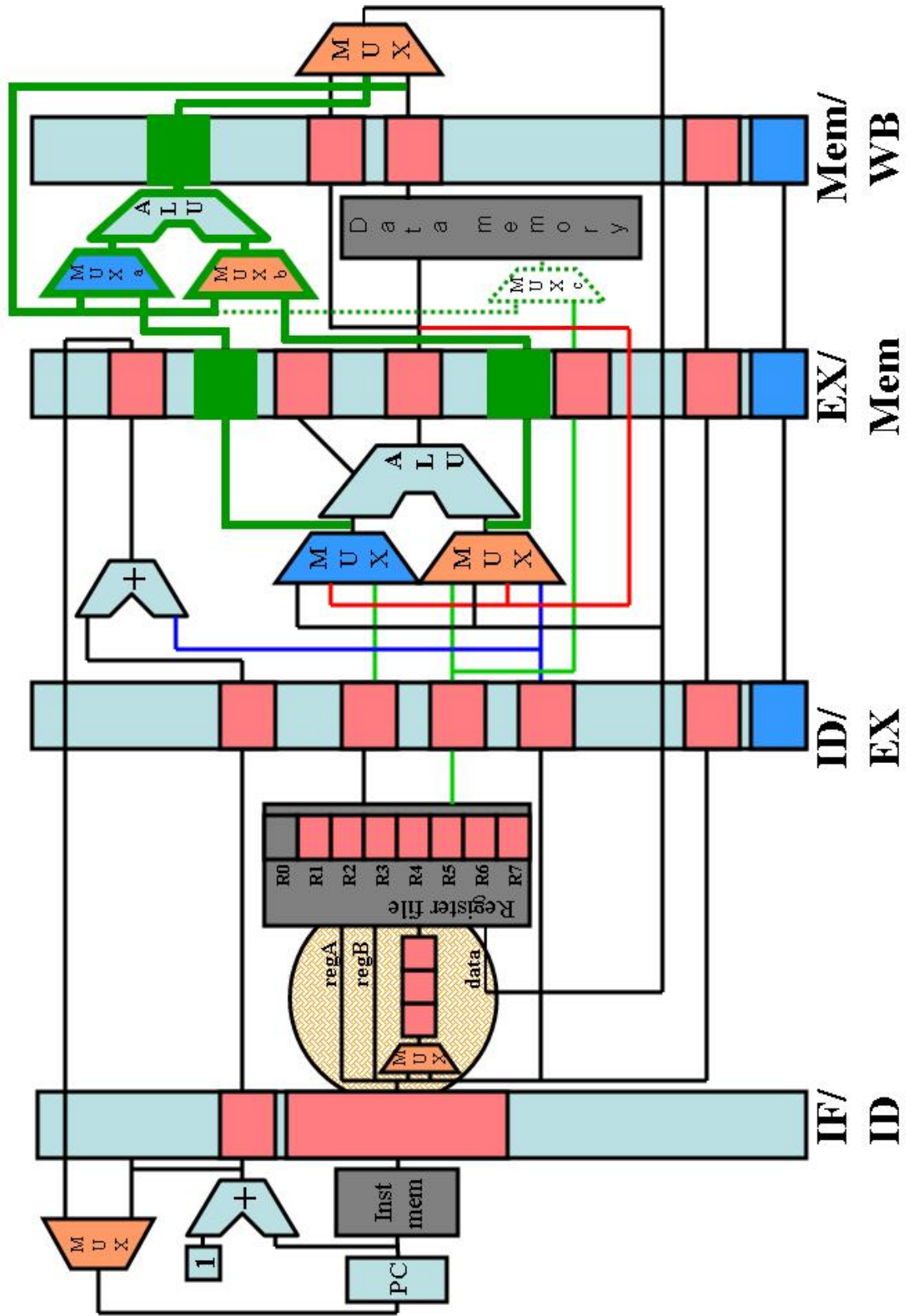
Follow these rules:

- You are allowed to expand multiplexers or pipeline registers, if needed.
- You are **not** allowed to modify the register file, the memory blocks, or the ALU.
- You are **not** allowed to increase the cycle time of the datapath.
- Assume that the register file, the ALU, and the memories have equal delays.
- Assume that registers, multiplexers, etc. have zero delay.

**Introduced hardware components, register fields, and wires are shown in dark green lines on next page. (Dotted components handle data hazards, in part, with I-type instructions. They are optional and not required for full credit.)**

b) [5 pts] For each hardware component that you added in your answer to part a), explain its role in the pipelined execution using the following table.

Component	Role
<b>ALU</b>	<b>Re-computes R-type operation (add, nand, etc.) in MEM stage using correct (original and/or forwarded) inputs.</b>
<b>MUX a</b>	<b>Allows selection between original input at ALU port A and forwarded data from MEM/WB register.</b>
<b>MUX b</b>	<b>Allows selection between original input at ALU port B and forwarded data from MEM/WB register.</b>
<b>Two register fields in Ex/Mem and one in Mem/Wb</b>	<b>Intermediate storage for forwarded regA, forwarded regB, and ALU result, respectively.</b>
<b>MUX c (optional)</b>	<b>Allows selection between original input at memory data port and forwarded data from MEM/WB register.</b>



c) [2 pts] Fill in the missing stages in the execution of the following instructions on your modified datapath.

	1	2	3	4	5	6	7	8
<b>lw 3 2 4</b>	IF	ID	Ex	Mem	WB			
<b>sw 5 2 6</b>		IF	ID	Ex	Mem	WB		

d) [2 pts] Is a stall required to execute the above pair of instructions? Why? (Write only one sentence.)

**1 stall cycle required, if dotted components are not introduced.**

**No stalling needed, if the dotted components are introduced.**

e) [2 pts] Now, fill in the missing stages in the execution of the following instructions on your modified datapath.

	1	2	3	4	5	6	7	8
<b>lw 1 5 3</b>	IF	ID	Ex	Mem	WB			
<b>sw 5 3 4</b>		IF	ID	ID	Ex	Mem	WB	

f) [2 pts] Is a stall required to execute the above pair of instructions? Why? (Write only one sentence.)

**1 stall cycle required.**

## 5. Short cache questions [15 points]

**Cache design.** Assume the following cache/memory design: 16-bit addresses, byte-addressable, cache size is 256 bytes, block size is 8 bytes, tag size is 11 bits.

a) [3 pts] What is the associativity of the cache?

Answer:

**8-way**

b) [3 pts] How many sets are in the cache?

Answer:

**4 sets**

**Block Size = 8 bytes =  $2^3$  bytes**

**Block Offset = 3 bytes**

**16-bit address – 11-bit tag – 3-bit block offset = 2-bit set index**

**Number of Sets =  $2^2$  sets = 4 sets**

**Cache Size = 256 bytes =  $2^8$  bytes**

**Number of Blocks =  $(2^8 \text{ bytes}) / (2^3 \text{ bytes/block}) = 2^5$  blocks**

**Associativity =  $(2^5 \text{ blocks}) / (2^2 \text{ sets}) = 2^3$  blocks/set  
= 8 blocks/set = 8-way set-associative**

**Cache access time.** Assume you have two cache designs, as described below:

<u>Cache A</u>	<u>Cache B</u>
1 MB L1 Cache	256 KB L1 Cache 768 KB L2 Cache
92% Hit Rate in L1	70% Hit Rate in L1 85% Hit Rate in L2
4 ns L1 access time	2 ns L1 Access Time 8 ns L2 Access Time
100 ns main memory access time	100 ns main memory access time

a) [3 pts] Calculate the average access time for Cache A.

$$AAT = (cache-access) + (miss-rate)*(memory-access)$$

$$AAT = (4 ns) + (0.08)*(100 ns) = \underline{12 ns}$$

or

$$AAT = (hit-rate)*(cache-access) + (miss-rate)*(cache-access + memory-access)$$

$$AAT = (0.92)*(4 ns) + (0.08)*(104 ns) = \underline{12 ns}$$

b) [5 pts] Calculate the average access time for Cache B.

$$AAT = (L1-access) + (L1-miss-rate)*[ (L2-access) + (L2-miss-rate)*(memory-access) ]$$

$$AAT = (2 ns) + (0.30)*[ (8 ns) + (0.15)*(100 ns) ] = \underline{8.9 ns}$$

or

$$AAT = (L1-hit-rate)*(L1-access) + (L1-miss-rate)*[ (L2-hit-rate)*(L1-access + L2-access) + (L2-miss-rate)*(L1-access + L2-access + memory-access) ]$$

$$AAT = (0.70)*(2 ns) + (0.30)*[ (0.85)*(10 ns) + (0.15)*(110 ns) ] = \underline{8.9 ns}$$

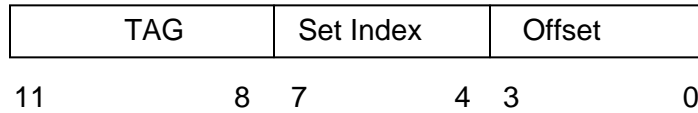
c) [1 pts] Which cache is faster? (circle one)

Cache A

Cache B

## 6. Cache simulation [15 points]

Assume a cache with the following configuration: total size is 1024 bytes, block size is 16 bytes, 4-way associative. The memory address size is 12 bits and is byte-addressable. Thus the partition of the address into tag, set index and offset is:



For this problem we want to consider a **MRU (most recently used) replacement policy**, since we believe that LRU is overrated, particularly with iterations over large arrays.

Consider the stream of load instructions with the memory addresses shown in the table below. For each address, indicate the set index, whether the reference is a hit or miss, and the type of a miss (compulsory, conflict, capacity):

Address	Set Index	Hit or Miss	Compulsory	Conflict	Capacity
0x2AF	A	Miss	X		
0x224	2	Miss	X		
0x235	3	Miss	X		
0x531	3	Miss	X		
0x83C	3	Miss	X		
0x8A1	A	Miss	X		
0x93A	3	Miss	X		
0x2A2	A	Hit			
0x33A	3	Miss	X		
0x220	2	Hit			
0x23A	3	Hit			
0x23B	3	Hit			
0x33C	3	Hit			
0x931	3	Miss		X	