University of California at Berkeley

College of Engineering

Computer Science Division --- EECS

Spring 1998 J. Wawrzynek

CS152 Computer Architecture and Engineering

Midterm II

Your Name: _____

ID Number: _____

This is a closed-book, closed-note exam. No calculators. You have 3 hours.

Each question is marked with its number of points (one point per expected minute of time).

Show your work. Write neatly and be well organized.

Good luck!

1. [30 points] Short answers. Please provide *short* answers to the following questions.

a. **[1 points]** True or False. Microprogramming is an effective controller design abstraction for RISC processors.

b. **[1 points]** "Vertical" microcode refers to encoded microcode. The unencoded version is referred to as _____.

c. **[1 points]** True or False. Superpipelining and Superscalar are both modern techniques for reducing CPI to less than 1.

d. **[1 points]** DRAM capacity multiplies by 4X approximately every _____ years.

e. **[1 points]** True or False. "Page mode" is a style of RAM that speeds access by eliminating refresh.

f. **[1 points]** True or False. A "write through" cache requires dirty bits.

g. **[1 points]** True or False. Average hard-disk seek times are on the order of 8-12us.

h. **[1 points]** In IO systems, device "polling" is a low-cost alternative to _____.

i. **[2 points]** Name the three "C"s of cache misses:

j. **[2 points]** What is the ideal speedup of a pipelined processor over a single cycle processor?

k. **[2 points]** List two techniques for handling data hazards.

l. **[2 points]** Sketch a typical curve showing latency versus percentage of maximum throughput for a IO system.

m. **[2 points]** Draw the transistor level circuit diagram for a CMOS dynamic RAM cell:

n. **[3 points]** Draw the transistor level circuit diagram for a CMOS static RAM cell:

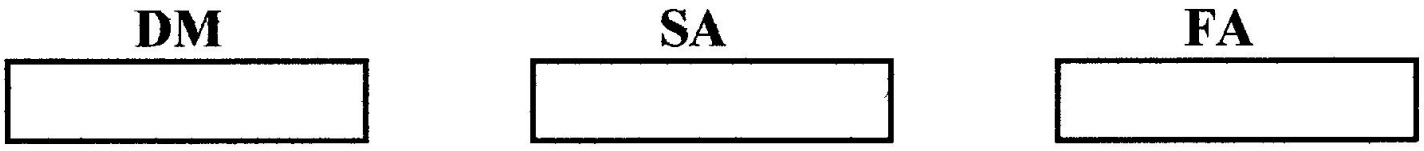
o. **[4 points]** Sketch a simple diagram showing a system that overlaps TLB and cache lookup. Draw a box for each of the CPU, TLB, Cache, and Main Memory.

p. **[5 points]** A processor has an ideal CPI=2.0 and executes with a clock cycle of 10ns. Assuming

a instruction mix including 30% load / store instructions, a unified cache hit rate of 90% and a miss penalty of 10 cycles, what is the real CPI?

2. [15 points] Consider the design of 3 different caches, one *direct mapped* (*DM*), one *2- way set associative* (*SA*), and one *fully associative* (*FA*). Each has 8-bit addresses (word addressing), 2 words / block, and a total capacity of (only) 8 words.

a. In the space below, for each type, label the address fields for the use as index and / or tag by the cache. Indicate the number of bits for each.



b. Assume initially the caches are empty (all invalid entries). For each of the read addresses listed in the table below, fill in a 0 or 1 for each cache type, indicating a hit (1) or a miss (0). Use a lest-recently-used (LRU) replacement policy.
c. Assuming a hit time of 1 cycle and a miss penalty of 20 cycles, compute the number of cycles for each cache to process the given address stream. Record you answer in the table.

address	0	2	4	6	8	0	2	4	6	8	1	2	3	4	5	10	# of cycles
DM																	
SA																	
FA																	

3. [20 points]

Consider the design of the processor with a 7-stage pipeline depicted below:

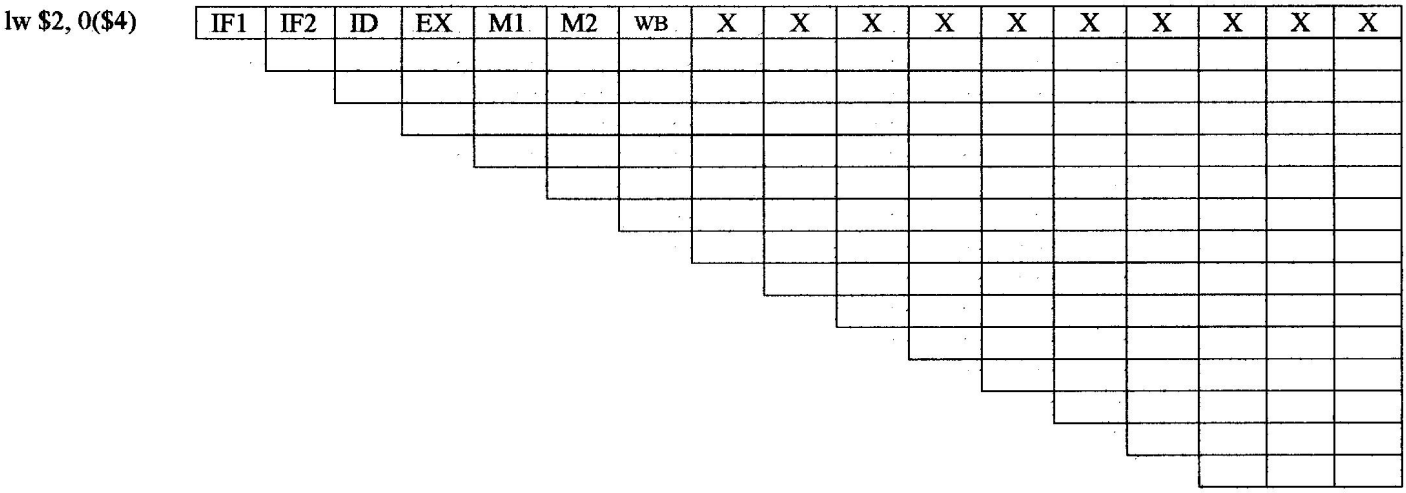
The pipeline is similar to the standard 5 stage pipeline with the exception that the icache and dcache are both pipelined with a latency of 2 cycles. As usual, the branch compare is calculated and the new PC loaded during the ID stage.

Assume the processor uses bypassing (forwarding) when necessary and possible.

a. Show the execution of one iteration of the following loop on the pipeline. Use the grid to illustrate the execution. In addition to one iteration, show the execution of one instruction from the next iteration of the loop, to illustrate when the next iteration would begin. As usual, instruction issue slots move from top to bottom and time from left to right.

Do not reorder the instruction.

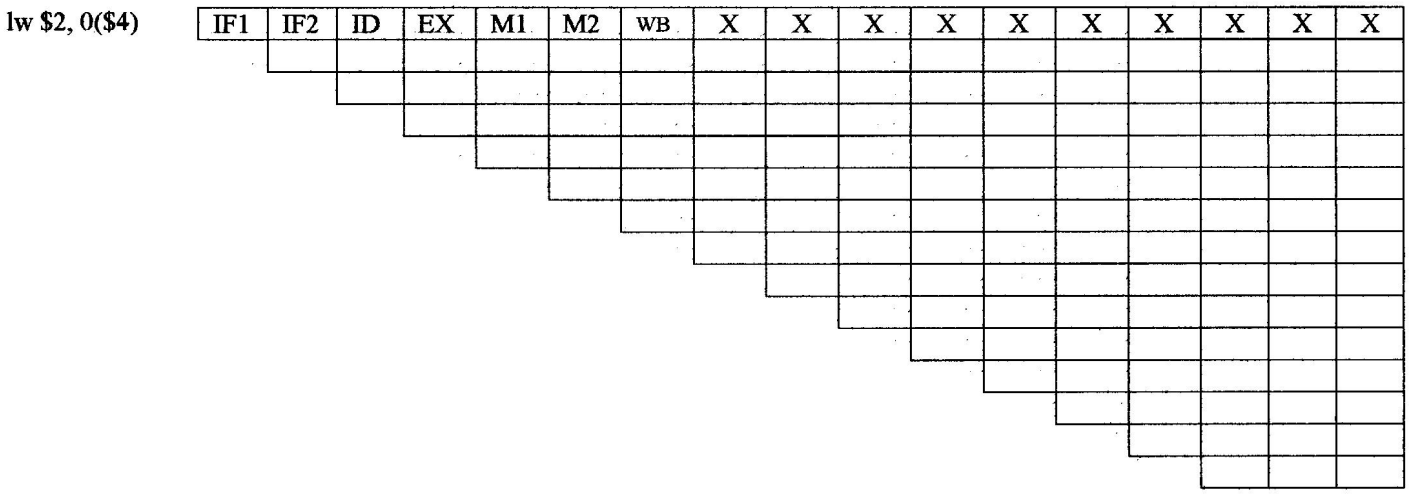
```
loop:  lw    $2,  0($4)
       add   $1, $1, $2
       add   $5, $5, $1
       addi  $4, $4, $4
       addi  $3, $3, -1
       bgez  $3, loop
       nop
       nop
       nop
```



Write down the number of instruction issue slots for one iteration here:

b. Now, assume that you can reorder instructions within one iteration of the loop in an attempt to decrease the CPI. However, your reordered loop must retain the same function as the original loop.

Show execution of your reordered loop below.



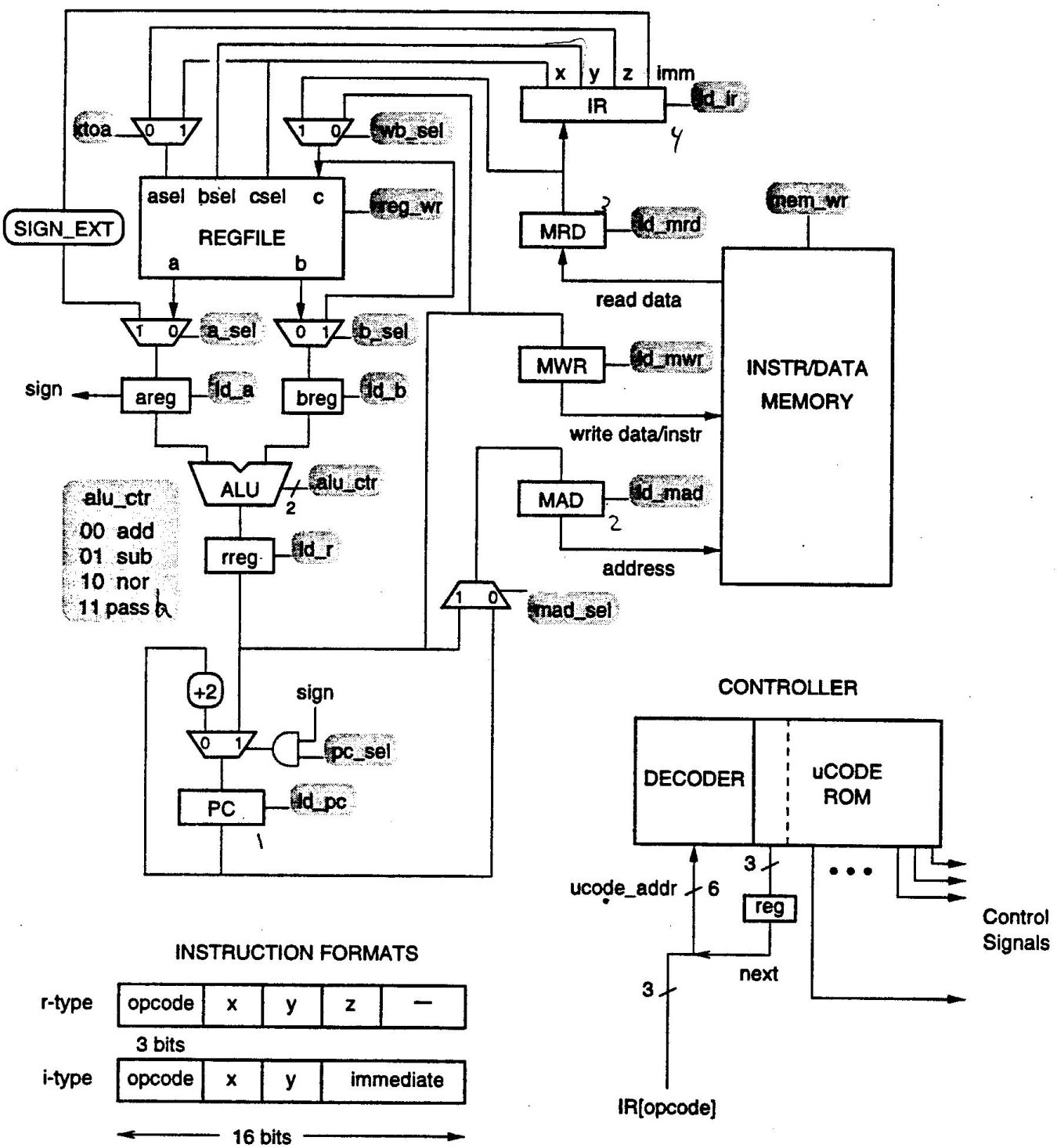
Write down the number of instruction issue slots for one iteration here:

c. Assuming that the loop executes for a very large number of iterations, what is the average CPI for each part? (Ignore the time it takes to fill the pipeline on the first iteration of the loop and the time it takes to drain the pipeline on the final iteration.)

Original order CPI =

Reordered CPI =

4. [35 points] This problem deals with a multi-cycle processor shown below:



Assume that a memory access can be completed in less than one cycle.

Each control signal is highlighted with a gray box. The signals beginning with "Id_" are register load enable signals and are active high. The write enable signals for the register file and the memory are also active high.

All corner signals are generated from a controller shown in the lower right hand corner. This is a microcode based controller with a section of microcode for each instruction. Each line of microcode includes a set of bits for driving the appropriate control signal and 3 bits used as the low-order bits of

the next microcode line. The 3 bit field is labeled "next". The high-order bits of the microcode address always come from the 3 opcode bits of the instruction register (IR).

For this problem we are interested in the instructions described in the table below, where "R", "M", "s_ext", and "imm", stand for "register file", "MEMORY", "sign_extend", and immediate respectively.

Instruction	Name	Description	Op
Logical NOR	NOR x,y,z	R[x] ← R[y] NOR R[z]	0
Load word	LDW x,y,imm	R[rx] ← M[R[y]+s_ext(imm)]	1
Store word	STW x,y,imm	M[R[y]+s_ext(imm)] ← R[x]	2
Branch < 0	BNEG x,y,imm	IF R[x] < 0, PC ← R[y]+s_ext(imm)	3
Add to Memory	ADDM x,y,imm	M[R[y]+s_ext(imm)] ← R[x] + M[R[y]+s_ext(imm)]	4

The table below represents the first 40 entries of the microcode ROM. The microcode line address is shown as a base-8 number. The first column of the table holds the 30bit "next" field. The remaining columns hold the state of the control signals.

Fill in the microcode table below for the above instructions. Use a base-8 number for the next column and 1's and 0's for the others. You do not need to fill in the 0 entries--any square left blank will be assumed to be a 0.

Try to minimize the number of cycles for each instruction, and don't forget instruction fetch.

microread address	next	xtoa	Wb_sel	Reg_wr	A_sel	B_sel	Id_a	Id_b	Alu_ctr[1]	Alu_ctr[0]	Ld_r	Pc_sel	Id_pc	Id_ir	Mem_wr	Id_mrd	Id_mwr	Id_mad	Mad_sel
00																			
01																			
02																			
03																			
04																			
05																			
06																			
07																			
08																			
09																			
10																			
11																			
12																			
13																			
14																			
15																			
16																			
17																			
18																			
19																			
20																			
21																			
22																			
23																			
24																			
25																			
26																			
27																			
28																			
29																			
30																			
31																			
32																			
33																			
34																			
35																			
36																			
37																			
38																			
39																			
40																			
41																			
42																			
43																			
44																			
45																			
46																			
47																			