Computer System Architecture
6.823 Midterm Examination
Spring 2002

Name:____________________

This is an open book, open notes exam.
110 Minutes
1 Pages

Notes:

- Not all questions are of equal difficulty, so look over the entire exam and budget your time carefully.
- Please carefully state any assumptions you make.
- The last 4 pages of the exam are for your reference only. Feel free to tear them off to look at. We will not grade anything you write on these pages.
- Please write your name on every page in the midterm (you get 5 points for doing this).

Name:	________	5 Points
Part A: (Question 1 - Question 2)	________	20 Points
Part B: (Question 3 - Question 4)	________	20 Points
Part C: (Question 5 - Question 7)	________	17 Points
Part D: (Question 8 - Question 10)	________	21 Points
Part E: (Question 11 - Question 15)	________	27 Points
Total:	________	**110 Points**

Name_________________________________

Part A: ISAs (20 points)

Question 1 (12 points)

Each of the following changes to a computer system may or may not cause existing user software to generate different results. State yes or no for each change, and explain.

Changing the cache line size?

Changing the number of levels in a hierarchical page table?

Converting a microcoded implementation to use nanocode?

Splitting the instruction fetch stage into more than one pipeline stage?

Adding more branch delay slots?

Adding new registers and new instructions that manipulate these registers? Assume the new instructions use previously illegal opcodes.

Name___________________________________

Question 2 (8 points)

Each of the following changes to a computer system may or may not cause the existing operating system software to generate different results. State yes or no for each change, and explain.

Changing the number of levels in a hierarchical page table?

Converting a microcoded implementation to use nanocode?

Splitting the instruction fetch stage into more than one pipeline stage?

Adding new registers and new instructions that manipulate these registers? Assume the new instructions use previously illegal opcodes.

Name________________________________

Part B: Microprogramming (20 points)

Question 3 (5 points)

We mentioned in tutorial that the microinstructions for JR and JALR in slide L4-20 (attached) were incorrect because it is not possible to perform the operation PC ← Reg[rf1] in one cycle. JR and JALR were corrected to the following:

JR0: A ← Reg[rf1]	JALR0: A ← PC
JR1: PC ← A	JALR1: Reg[31] ← A
	JALR2: A ← Reg[rf1]
	JALR3: PC ← A

This is still not entirely correct. Explain what can go wrong and write the correct microinstruction sequences for JR and JALR.

Question 4 (15 points)

In this question, you will implement SLL (shift left logical) on the microcoded DLX machine. A diagram of the microcoded DLX datapath is attached at the back of the exam.

SLL can be described by the following semantics:

```
SLL Rd,Rs1,Rs2       # Rd <- Rs1 << Rs2
```

The contents of `Rs1` are shifted left, inserting zeroes into the emptied bits, and the result is placed into `Rd`. The bit shift count is specified by `Rs2`. `Rs1` and `Rs2` are unsigned 32-bit integers and the format of SLL is R-type. (Hints: Adding a number to itself is equivalent to left-shifting the number once. Use `Rd` as a counter.)

The possible ALUOps are COPY_A, COPY_B, INC_A_1, DEC_A_1, INC_A_4, DEC_A_4, ADD, and SUB. The possible μBr (microbranch) values are N (next), J (jump), Z (branch-if-zero), and D (dispatch).

Fill out the table to implement SLL. Optimize your microcode. Part of the table has already been filled out.

State	RTL Code	ALUOp	μBr	Next State
FETCH0	MA ← PC	*	N	*
FETCH1	IR ← Mem	*	N	*
FETCH2	A ← PC	*	N	*
FETCH3	PC ← A + 4	INC_A_4	D	*
...				
SLL0	A ← Reg[rf2]	*	N	*
SLL1	B ← Reg[rf1]	COPY_A	Z	
SLL2	Reg[rf3] ← A			
	Reg[rf3] ← B	COPY_B	J	FETCH0

Name___________________________________

Part C: Pipelining (17 points)

Question 5 (2 points)

Machine A has a 5-stage pipeline, runs at 500 MHz, and has an average CPI of 1.5.
Machine B has a 20-stage pipeline, runs at 2 GHz, and has an average CPI of 2.0.

Which machine has the higher performance? How much faster is it than the other machine?

Question 6 (10 points)

Fill in the resource usage diagram for execution of the following program. Assume instructions are being executed on a 5-stage fully bypassed DLX datapath. A diagram of this datapath is attached at the back of this exam. Jumps and branches are resolved in ID. Take into account that the DLX has one delay slot. Assume memory is made of MAGIC RAM.

```
L1: LW    R1,0(R20)
    ADDI  R0,R1,#1
    SUBU  R1,R0,R0
    JAL   L2
    LW    R6,4(R20)
    BEQZ  R1,L3
    AND   R6,R5,R4
    ANDI  R9,R8,#1
L2: ADD   R20,R6,R8
    JR    R31
    SGT   R8,R9,R10
    SLT   R0,R8,R7
L3: ADDU  R12,R11,R10
```

	t2	t3	t4	t5	t6	t7	t8	t9	t10	t11	t12	t13
IF	ADDI											
ID	LW											
EX												
MA												
WB												

Question 7 (5 points)

The following sequence of DLX instructions raises the exceptions shown:

`LW`	`R2,4(R3)`	data TLB miss, page fault on data memory
`ADD`	`R4,R5,R6`	overflow
`SW`	`R7,8(R8)`	instruction TLB miss, protection violation on data memory
`???`		invalid opcode

Assume that exceptions are precise, and are handled in the memory stage of the 5-stage DLX pipeline (as discussed in lecture 10, slide 25 - attached). During execution, there are no pipeline bubbles between these instructions in the pipeline.

Which of the above exceptions is detected first by the pipeline control logic?

Which of the above exceptions causes the first jump to the interrupt handler?

Part D: Cache Performance (21 points)

For these questions, we will be analyzing the performance of two different caches with the following parameters.

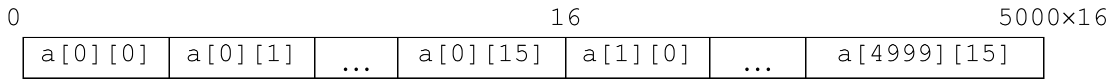
	Cache A	Cache B
Associativity	Direct Mapped	Fully Associative
Block Size	4 words	4 words
Cache Capacity	4 blocks	4 blocks
Replacement	Replace line	LRU
Write Policy	Write through/ No write-allocate	Write through/ No write-allocate

Consider the following program written in pseudo-code:

```
for (i = 0; i < 5000; i++) {
     sum = 0;
     for (j = 0; j < 16; j++) {
          sum = sum + a[i][j] * h[j];
          /* load a[i][j] before h[j] in assembly */
     }
     b[i] = sum;
}
```

Assume the following memory layout of each array (addresses used in MEM[] are word addresses).

```
a[i][j]: MEM[16*i+j]
```



```
b[i]: MEM[16*5000+i]
h[j]: MEM[16*6000+j]
```

For the problems below, assume that the caches are initially empty. Also, assume that only accesses to the arrays (‘a’, ‘h’, and ‘b’) cause memory references (all other necessary variables are stored in registers).

Name________________________________

Question 8 (8 points)

Calculate the number of misses that will occur when running the program for Cache A. Note that the cache has a no-write-allocate policy.

The number of misses for Cache A: ____

Calculate the number of misses for Cache B.

The number of misses for Cache B: ____

Which cache will perform better?

Name________________________________

Question 9 (8 points)

We consider a software approach to remove conflicts between array '`a`' and '`h`'. We exploit the fact that we can control the data placement in the direct-mapped cache by carefully choosing the memory layout. We change the memory layout of array '`h`' as follows:

```
h[j]: MEM[16*6000+j+4]
```

Calculate the number of misses that will occur when running the program with the new memory layout for Cache A.

The number of misses for Cache A: ________________________________

Calculate the number of misses for Cache B.

The number of misses for Cache B: ________________________________

Which cache will perform better?

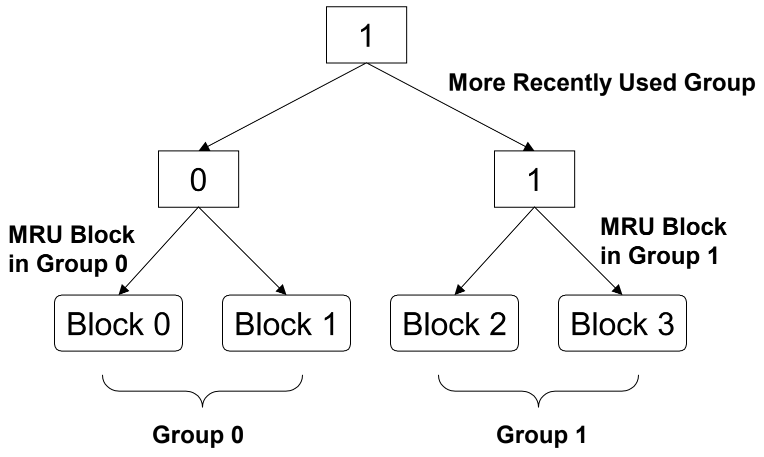
Name__________________________________

Question 10 (5 points)

The fully-associative cache uses an accurate LRU replacement policy. Now, consider the two-level pseudo LRU replacement policy shown in the following figure.

Cache blocks within a set are divided into two groups and three pseudo LRU state bits are added to each cache set. One bit indicates which group is more recently used. If the bit is 1, Group 1 is more recently used. Then, within each group, another bit indicates which cache block is more recently used within the group. If this bit is 1, the cache block on the right side is more recently used. The less recently used block in the less recently used group is evicted on a cache miss (In the figure, block 1).

Provide a short sequence of memory accesses using word addresses (for example, 0 4 8 12 16 0 4 8 12 16) that will cause the pseudo LRU policy to replace a different cache block than the accurate LRU policy. Start the sequence with "0 4 8 12".

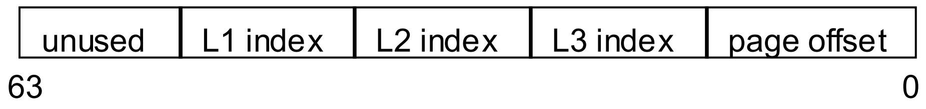


Address sequence: 0 4 8 12

____

Part E: Virtual Memory (27 points)

Consider a byte-addressable machine with a virtual memory system that has 64-bit virtual addresses and 4KB pages. The page table is a three-level hierarchical page table, where L1, L2, and L3 indices are 12 bits each. As in problem 4 of problem set 3, only part of the virtual address space is implemented; the topmost bits of the virtual address are not used to index the virtual memory. Before the virtual address is sent to the virtual memory system, a special hardware circuit checks that the topmost unused bits are either all zeroes or all ones and match the top bit of the L1 index.



Question 11 (2 point)

How many entries are there in the level 1 page table?

Question 12 (2 points)

What is the size of the part of the virtual address space that is implemented?

Question 13 (15 points)

The diagram on the following page shows fragments of the contents of a three-level, hierarchical page table for the machine described. Page table entries in level 1 and level 2 page tables contain the physical address or disk block number of the next level table. Page table entries in the level 3 tables contain the physical page number or disk block number of the requested page. Physical addresses are 28 bits.

The size of a page table entry is 4 bytes and the root of the current page table is at memory address `0x0004000`. (The "`0x`" prefix indicates that the address is expressed in hexadecimal format.) Lower addresses correspond to lower page table indices. Only valid page table entries are shown; you may assume that any page table entry not shown is invalid. The "R" column corresponds to the resident bit as described in problem 3 of problem set 3.

What is the result of looking up the following virtual memory addresses in the page table shown? For each virtual address, circle the correct answer, and fill in the corresponding blank, if any.

A hexadecimal to binary conversion table is provided on the next page for your convenience.

Page table walk for virtual address `0x0000004010110804` results in:

a) Page table entry invalid exception
b) Page fault on page table entry
c) Page fault, disk block number ___________________
d) Resident, physical address ___________________

Page table walk for virtual address `0x00000001113B0110` results in:

a) Page table entry invalid exception
b) Page fault on page table entry
c) Page fault, disk block number ___________________
d) Resident, physical address ___________________

Page table walk for virtual address `0x0000001101002CD0` results in:

a) Page table entry invalid exception
b) Page fault on page table entry
c) Page fault, disk block number ___________________
d) Resident, physical address ___________________

Name___________________________________

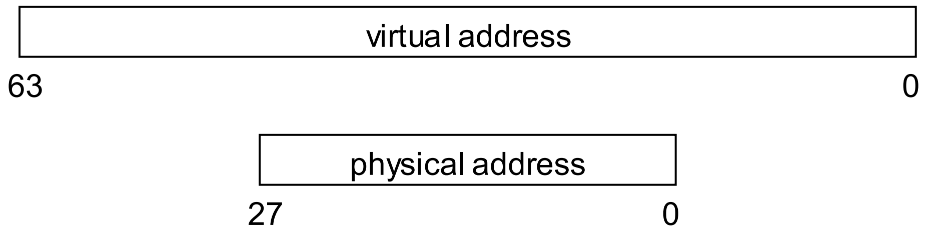
	PA/PPN/DBN	R
0x001C450	0x000100B	0
0x001C44C	0x000100D	0
0x001C448		
0x001C444	0x000100E	0
0x001C440	0x0020000	1
0x0014440	0x0110	1
0x001443C	0x0100	1
0x0014438		
0x0014434	0x0108	1
0x0014430		
0x0010050		
0x001004C		
0x0010048		
0x0010044	0x000100C	0
0x0010040	0x0014000	1
0x000C010	0x0104	1
0x000C00C		
0x000C008	0x000100A	0
0x000C004		
0x000C000	0x010C	1
0x0008410	0x0024000	1
0x000840C		
0x0008408		
0x0008404	0x000C000	1
0x0008400		
0x0004010	0x0010000	1
0x000400C		
0x0004008		
0x0004004	0x0008000	1
0x0004000	0x001C000	1

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Name__________________________________

Question 14 (3 points)

Our machine also has a single cache, which is accessed in parallel with a 64-entry fully-associative TLB. The cache is a virtually-indexed, physically tagged, 16KB direct-mapped cache, with 4-word cache blocks. The size of a word is 4 bytes. Keep in mind that the topmost bits of the virtual address are not used by the virtual memory system.



Which bits of the 64-bit virtual address are translated by the TLB lookup? _____:_____
Which bits of the 64-bit virtual address are needed to index into the cache? _____:_____

Which bits of the 28-bit physical address form the cache tag? _____:_____

Question 15 (5 points)

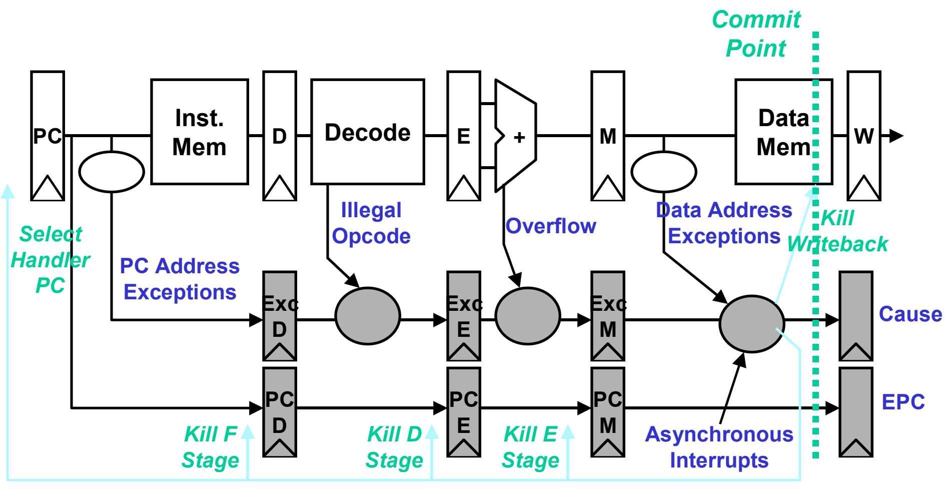
In our system, we prevent aliasing as follows. If two virtual addresses map to the same physical address, we require that the operating system make the page offsets of the virtual addresses agree. Does this scheme work? Explain.

Name______________________________________

Lecture 4-20

State	Control points	next-state
J_0	A ← PC	next
J_1	B ← $sExt_{26}$(Imm)	next
J_2	PC ← A+B	fetch
JR_0	PC ←Reg[rf1]	fetch
JAL_0	A ← PC	next
JAL_1	Reg[31] ← A	next
JAL_2	B ← $sExt_{26}$(Imm)	next
JAL_3	PC ← A+B	fetch
$JALR_0$	A ← PC	next
$JALR_1$	Reg[31] ← A	next
$JALR_2$	PC ←Reg[rf1]	fetch

Lecture 10-25



Name___________________________________

A Bus-based Datapath for DLX

Name____________________________________

Fully Bypassed Datapath

Name_______________________________

Part F: Midterm Summary