Name: _____

Email address: _____

# CSE 378 Autumn 2008: Final Exam

(closed book, closed notes, NO calculators allowed)

**Instructions:** Read the directions for each question carefully before answering. We may give partial credit based on the work you **write down**, so if time permits, show your work!

Good Luck!

Total: 90 points. Time: 110 minutes.

| Page # | Max Points Possible | Page Score |
|---|---|---|
| 2 | 8 | |
| 3 | 15 | |
| 4 | 12 | |
| 5 | 12 | |
| 6 | 13 | |
| 7 | 14 | |
| 8 | 16 | |
| ***Total*** | 90 | |

There is an ***extra credit*** question on the back of the last page of the exam. If you tear off the last page, please write your answer here:

1. [8 points] **Optimization/Locality**

Assuming `ints` are 4 bytes (32 bits), cache lines (blocks) are 16 bytes, and the following declarations in C:

```
int x[MAX1][MAX2];
int y[MAX2][MAX1];
int i, j;
```

A programmer takes this original code:

```
// old code...
for (j = 0; j < MAX2; j++)
     for (i = 0; i < MAX1; i++)
          x[i][j] = x[i][j] + y[j][i];
```

And modifies it to look like this:

```
// ...new code
for (i = 0; i < MAX1; i++)
     for (j = 0; j < MAX2; j++)
          x[i][j] = x[i][j] + y[j][i];
```

a) With this change the new code has been *improved* in which of the following ways (circle **all** that apply) :

1) Take better advantage of temporal locality on x
2) Take better advantage of temporal locality on y
3) Take better advantage of temporal locality on i
4) Take better advantage of temporal locality on j
5) Take better advantage of spatial locality on x
6) Take better advantage of spatial locality on y
7) Take better advantage of spatial locality on i
8) Take better advantage of spatial locality on j

b) Would you recommend a **write through** or a **write back** cache for the new code (circle **one**)?

c) Explain your choice in b).

2. [15 points] **Memory**

Assuming you have:

16 MB Physical Address Space
4 GB Virtual Address Space
8 KB Page size

Answer the following questions. *If you do not have enough information to answer the question, say* "**Not Enough Info**".

a) How many **bits** are needed to specify the Page Offset?

b) How many **bits** are needed to specify the Physical Page Number?

c) How many **bits** are needed to specify the Virtual Page Number?

d) How many TLB **entries** (total)?

e) How many Page Table **entries** (total)?

3. [12 points] **Caches/Virtual Memory**

Assuming you have:

32-bit addresses (virtual and physical)
1 KB Page size
4 KB, 2-way set-associative, write through cache, LRU replacement
(the cache holds 4 KB total of data, do not count other fields in this count)
64 Byte cache block (line) size
4-entry fully associative TLB

Answer the following questions. *If you do not have enough information to answer the question, say* "**Not Enough Info**".

a) Number of *bits* needed to specify **byte offset** in a cache line: _____

b) Number of *bits* needed to specify the **tag**: _____

c) Number of *bits* needed to specify the **index**: _____

d) **Total** number of *blocks* (lines) in the cache: _____

4. [12 points] **Caches/Virtual Memory**

***Given the cache described in the previous problem (problem 3)*** and the following C code (assume `ints` are 4 bytes):

```
#define MAX 2048
int A[MAX];    // Assume A starts at address 0x1000
int i, sum = 0;

for (i = 0; i < MAX; i++) {   // 1st loop starts here
     A[i] = i;
}
for (i = 0; i < MAX; i += 32) { // 2nd loop starts here
     sum = sum + A[i];
}
```

Answer the following questions. Ignore variables `i` and `sum` for this question (focus only on array A). Assume no context switches. Assume the two loops run in the order shown above. You may give answers as fractions or percentages. *If you do not have enough information to answer the question, say* "**Not Enough Info**".

a) What is the hit rate for the cache in the *first* loop: _____

b) What is the hit rate for the cache in the *second* loop: _____

c) What is the hit rate for the TLB in the *first* loop: _____

d) Number of page faults in the *second* loop: _____

5. [8 points] **Performance:** Given the following mix of instructions in a set of benchmarks we are interested in, and CPI for each instruction type:

| Instruction Type | Frequency | CPI |
|---|---|---|
| A | 25% | 1 |
| B | 10% | 5 |
| C | 35% | 3 |
| D | 30% | 4 |

a) What is the overall CPI?

b) If we could figure out a way to make type D instructions take 0 cycles each, how much faster could we run the benchmarks?

6. [5 points] **I/O**: Assume a hard disk has the following specifications:

   - An average seek time of 5 ms
   - A 6000 RPM rotational speed
   - A 20 MB/s average transfer rate
   - 3 ms of overheads for making a request

Given that sectors are 512 bytes, ***write an equation using the values given above*** that represents the average time to read or write a 512-byte sector? (You are not required to solve the equation.)

Average time
to read/write =
a sector

7. [4 points] **I/O:** Why might **interrupts** be preferable to **polling** for communicating with I/O devices?

8. [5 points] **MIPS**: Function A calls function B. Function B calls function C. Function C does not call any functions.

   - Function A cares about the values it has stored in registers $t0 and $t1 (and needs to use them after returning from the call to B)..
   - Function B uses registers $t0 and $t1 but is done using them (and no longer cares about their values) before calling function C.
   - Function C uses registers $t0 and $t1.

   Who, if anyone should save registers $t0 and $t1? (circle **all** that apply)

   a) Function A

   b) Function B

   c) Function C

9. [5 points] Indentify the *true(flow)* **data dependences** in the following MIPS code (**draw arrows** from *operand* to *operand* to show how the values flow):

```
addi    $6, $6, 1

sub     $5, $6, $12

srl     $5, $5, 4

lw      $5, 0($5)

sw      $5, 0($5)

beq     $6, $18, foo
```

10. [16 points] **Pipelining**: These questions refer to the pipelined processor shown on the last page of the exam (feel free to tear the page off).

a) [5 points] *For the* processor *shown on the last page*, consider executing the following *data-dependence-free* MIPS code:

```
add  $1, $2, $3
sub  $4, $5, $6
sub  $7, $8, $9
add  $10,$11,$12
add  $13,$14,$15
sub  $16,$17,$18
```

During the **sixth** cycle of execution, which registers (in the register file) are being **read** and which register will be **written**?

**Writing:** _____ **Reading:** _____

b) [6 points] *For the* processor *shown on the last page*, and the following snippet of code:

```
lw       $2, 0($6)
add      $8, $2, $3
```

(Note: This question is not about the memory hierarchy, it refers to the processor only.)

*(1) If we do not use forwarding at all*, how many cycles will we need to stall for? (**Circle** your answer)

**0 1 2 3 4 5**

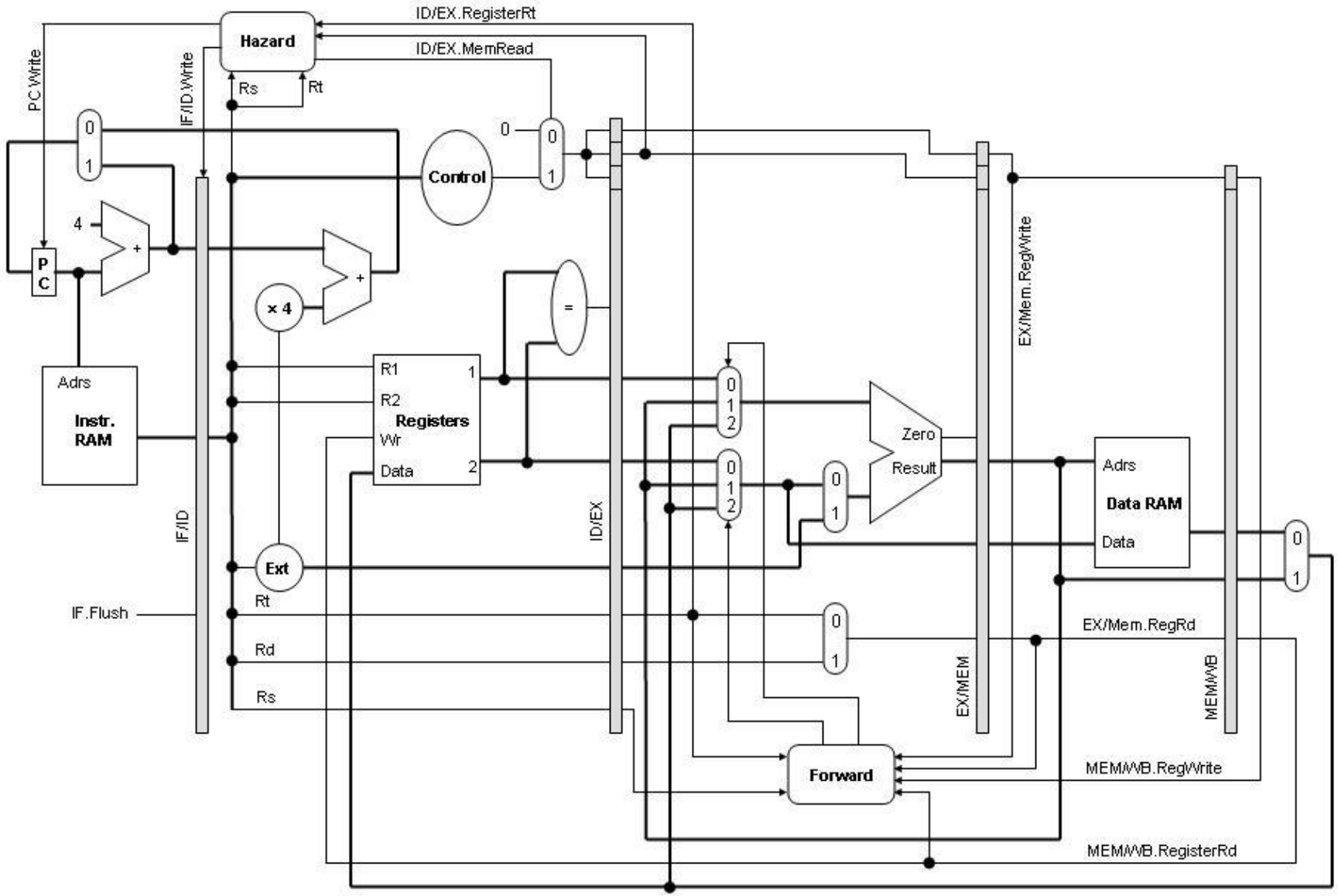
*(2) If we use the forwarding shown for this processor*, how many cycles will we need to stall for? (**Circle** your answer)

**0 1 2 3 4 5**

c) [5 points] *For the* processor *shown on the last page*, finish the load hazard detection equation for the stall condition in b) above: (be sure to get your parentheses correct)

**if ( (ID/EX.MemRead = 1)**

**then stall**


ID/EX.RegisterRt
Hazard
ID/EX.MemRead
PCWrite
IF/ID.Write
Rs
Rt
0
1
Control
0
4
+
P
C
× 4
=
Adrs
Instr.
RAM
R1
R2
Registers
Wr
Data
1
2
IF/ID
ID/EX
Ext
IF.Flush
Rt
Rd
Rs
0
1
2
Zero
Result
EX/Mem.RegWrite
Adrs
Data RAM
Data
EX/Mem.RegRd
EX/MEM
MEM/WB
Forward
MEM/WB.RegWrite
MEM/WB.RegisterRd

Extra Credit Question:

When running a program that has been parallelized with fork-join parallelism on P processors, the user does not see a speedup of P. Give **two reasons** why this is possible.