

# CS 473

## Final Exam

### December 10, 2004

### Solutions

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no electronic aids** are allowed. Please note the following instructions. There will be a ten point deduction for failure to comply with them:

- start each problem on a new sheet of paper
- write your social security number, but not your name, on each sheet of paper you turn in
- show your work whenever appropriate. There can be no partial credit unless I see how answers were arrived
- be succinct. You may lose points for facts that, while true, are not relevant to the question at hand

You have until 3:00 to finish the exam.

1. (20 points) Floating point arithmetic:

(a) Add the following pair of IEEE floating point numbers:  $0x40480000 + 0xc0d80000$

i. Break into components:

	40480000	c0d80000
Binary	0100 0000 0100 1000	1100 0000 1101 1000
Sign	0	1
Exponent	10000000	10000001
Significand	1.1001	1.1011

ii. Denormalize and (since signs differ) subtract:

$$\begin{array}{r}
 1.10110 \\
 - .11001 \\
 \hline
 .11101
 \end{array}$$

iii. Renormalize and combine

Sign	1
Exponent	10000000
Significand	1.1101
Binary	1 10000000 1101
Hexadecimal	c068

Solution:  $0xc0680000$

Forgot phantom bit	-3
Forgot to denormalize	-3
Subtracted exponents	-3
Seven bit exponent	-1
Field in hex version doesn't match fields (no binary version)	-2
Subtracted backwards	-2
Took exp from larger instead of renormalizing correctly	-1
Tried to use 2's comp addition to subtract (but wrong)	-3

(b) Convert your result into human-readable decimal.

(working from table at end of Part 1a):

Denormalize and find integer and fraction parts:  $-11.101_2$ .

Convert to decimal:  $-3.625$

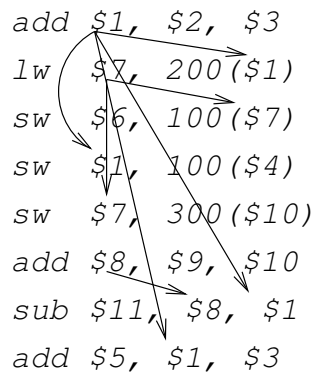
Correct sign in binary result, forgot in decimal	-1
Interpreted exponent very, very wrong	-2
Left-shifted for negative exponent	-2
Right-shifted for positive exponent	-2
Didn't denormalize	-2
Fraction part close but wrong, no work shown	-2

## 2. (40 points) Superscalar Pipelining

Consider the following sequence of MIPS instructions:

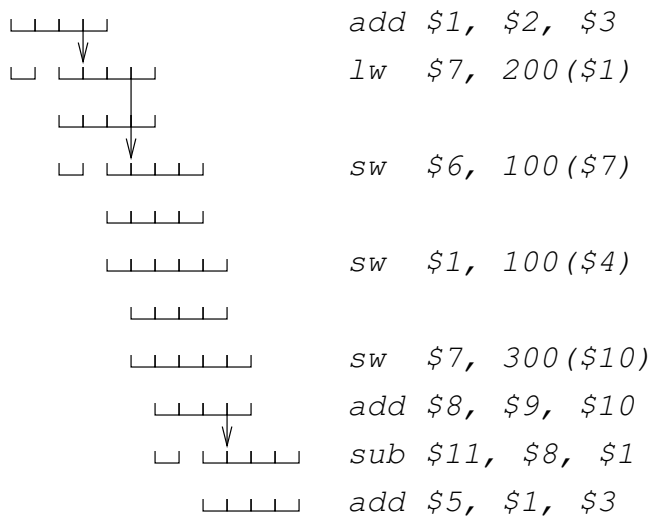
```
add $1, $2, $3
lw $7, 200($1)
sw $6, 100($7)
sw $1, 100($4)
sw $7, 300($10)
add $8, $9, $10
sub $11, $8, $1
add $5, $1, $3
```

(a) Draw arrows showing all of the dependencies between the registers in instructions in this code. Note that if a register is set by an instruction, and then read by two different instructions, there are two dependencies.



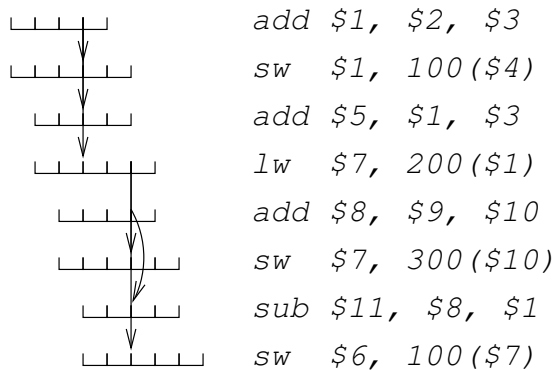
Each missed or extra dependency | -1

(b) Draw a Gantt chart showing how this code is executed in the superscalar pipeline from HW 5.



Failure to stall next instruction when earlier stalls | -1  
 Five stage top pipe | -3  
 Only one pipeline | -5

(c) Reorder the code to execute as quickly as possible and draw another Gantt chart showing how the modified code is executed.



Each extra stall | -2  
 Failure to stall next instruction when earlier stalls | -1  
 No Gantt chart | -10  
 No Part C at all | -20  
 dependency violation | -3

### 3. (15 points) Cache

(a) A computer with a 32-bit address has a 64K, 16-way set-associative cache with a 16 byte block size. What are the tag, index, and byte offset for address 0x12345678?

Since the block is 16 bytes, the byte offset is 4 bits.

Since the total cache size is 64K and the block size is 16 bytes, there are  $64K/16 = 4K$  blocks in the cache. As it's 16-way set-associative, there are  $4K/16 = 256$  sets. The index is  $\log_2 256 = 8$  bits.

This leaves  $32 - 4 - 8 = 20$  bits of tag.

The given address breaks down as:

Tag	12345
Index	67
Offset	8

Incomprehensible Tag Calculation | -3  
 Wrong Index | -3  
 Calculated field sizes, didn't break down address | -2  
 Direct-mapped | -3

- (b) A computer has a 13 bit index and 5 bit byte offset. If it uses a 32 bit address, what is the width of the tag field? If it has a 512K cache, how set-associative is it?

*The tag field is  $32 - 13 - 5 = 14$*

*Since it has a 13 bit index, it has  $2^{13} = 8K$  sets. Since the byte offset is 5 bits, it has a 32 byte block. The size of the cache is 512K, so it contains  $512K/32 = 16K$  blocks. Therefore, it is  $16K/8K = 2$ -way set-associative.*

*Wrong associativity, don't know how they calculated it* | -5  
*32K blocks (how?)* | -3

#### 4. (20 points) Virtual Memory

Following are the PDBR and some memory addresses from an Intel. What is the result of each of the following memory accesses (all the numbers are hexadecimal)?

- (a) Kernel mode write to 097e8b68

*Directory entry is at 3adcc094, containing 2e673006. Present bit is 0, so page fault.*

- (b) user mode read from 17e289f0

*Directory entry is at 3adcc17c, containing 4f244003. U bit is 0, so protection violation.*

- (c) User mode write to 5957d91c

*Directory entry is at 3adcc594 containing 70b29007. Protection bits OK, so proceed to page table entry at 70b295f4 containing 19af3007. Protection bits OK again, so write to physical address 19af391c.*

PDBR: 3adcc000

Address	Contents
19af391c	47c476d9
2e673fa0	44aed006
2fa4d9f0	4e01295f
3adcc094	2e673006
3adcc17c	4f244003
3adcc594	70b29007
44aedb68	1270734d
4f2448a0	2fa4d003
70b295f4	19af3007

*Translated wrong, interpreted no data as miss* | -7  
*Didn't see protection failure* | -3  
*Took 47cf76d9 as final address to write to* | -2  
*Took address of entry instead of contents for protection bits* | -3

#### 5. (5 points) Input/Output

Suppose a computer has an IO subsystem that has a not-PCI but very convenient 100 MHz, capable of performing a 4-byte transfer on every bus cycle. Address and data are multiplexed (so every transfer requires one address cycle followed immediately by several data cycles).

- (a) What is the theoretical maximum amount of data that can be transferred per second (the throughput), given an unlimited number of data cycles in a burst?

*$100MHz \times 4 = 400MB/sec = 3200Mb/sec$*

- (b) What is the actual amount of data transferred per second if each transfer has exactly one four-byte data cycle?

*Since transfers will be alternating addresses and data, the throughput is cut in half ( $200 MB/sec = 1600Mb/sec$ ).*