

CS 473

Midterm Exam

October 16, 2002

The following exam is open book and open notes. You may feel free to use whatever additional reference material you wish, but **no calculators** 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 10:30 to finish the exam.

1. (30 points) Convert the following number from human-readable decimal to IEEE format: -12.375 (express your result in hexadecimal). Use the floating point addition algorithm to add it to the IEEE floating point number 0x40200000 (once again, express your result in hexadecimal). Convert your final result to human-readable decimal.

```

12 => 1100
.375 => .011
1100.011 => 1.100011*2^3
127 + 3 = 130 => 10000010
1 10000010 1000110-0
1100 0001 0100 0110 0-0
c1460000
40200000
0100 0000 0010 0-0
0 10000000 0100-0
Positive
exponent 10000000
significand 1.01
Denormalize to line up exponents:
.0101
Subtract because signs are different:
1.100011
-.0101
-----
1.001111
No need to renormalize.
1 10000010 0011110-0
1100 0001 0001 1110 0-0
c11e0000
Converting back to decimal, we have
-1001.111
-9.875

```

Points:

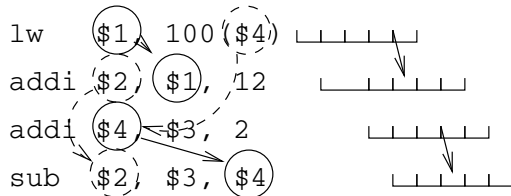
- ten points for initial conversion, ten for subtraction, ten for final conversion

<i>Penalty</i>	<i>Error</i>
-3	<i>exponent change with hidden bit</i>
-3	<i>forgot to shift to match exponent</i>
-1	<i>wrong sign on result</i>
-2	<i>subtracted larger from smaller</i>
-3	<i>forgot to denormalize</i>
-3	<i>added instead of subtracting</i>

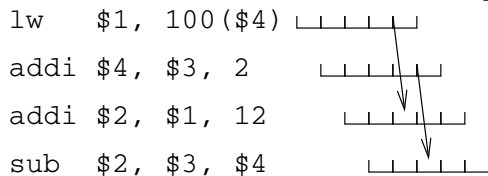
2. (20 points) Consider the following MIPS code fragment:

```
lw    $1, 100($4)
addi  $2, $1, 12
add   $4, $3, 2
sub   $2, $3, $4
```

- Draw arrows showing the dependencies in the code.
- Draw dashed arrows showing the antidependencies.
- Draw a Gantt (timing) chart showing the execution of this code, assuming all possible forwarding and assuming loads stall instead of using a delayed load. Use arrows to show forwarding between the instructions.



- Reorder the code so that it can run as fast as possible.



(I realize I didn't ask you to draw the Gantt chart this time)

Points:

- 5 points per part

Penalty	Error
-3	Missed (anti)dependency
-1	Extra forward
-1	Dependency for antidep
-3	No stall

- (30 points) The book has a possible superscalar version of the MIPS on page 512. I've copied this figure on the last page of this exam, for use in problem 3a

- In the figure as given, the top pipeline takes just as long to execute as the bottom pipeline. How could the top pipeline be modified to execute in only four cycles, instead of five?

There are a variety of ways to do this; basically, it has to bypass either the EX/MEM or the MEM/WB pipeline register.

- After making the modification described in part 3a, suppose the following two instructions are in the pipelines (so the first instruction is in the top pipe, and the second is in the bottom):

```
ori  $1, $2, 0x100
lw   $1, 100($2)
```

What will happen to register \$1?

Basically, the result will be the same as for the previous version of the pipeline: the lw ends up setting the new value of the register. You can phrase this in terms of the ori setting it and then the lw changing it, or the ori's register write-back getting cancelled.

- Now suppose the following two pairs of instructions are in the pipeline:

```
add  $3, $4, $5
lw   $1, 100($2)
ori  $1, $2, 0x100
sw   $6, 100($7)
```

Describe the hazard that is created. How can it be resolved?

It's a structural hazard, as the lw and the ori will try to write the register simultaneously (I'll accept an answer that calls it a WAW hazard on writing to the register, as well). Notice this same structural hazard actually existed for the original version of the pipelines, in the previous code sequence.

The lw's store has to be cancelled in favor of the ori.

Points:

10 points per part

Points	Error
-5	Also tried to shorten bottom pipe
-3	Stall ori in last part
-5	Cut off two stages
-5	Gantt chart correct, said same time
-10	Thought ori depended on lw

1. (20 points) Draw a Gantt chart showing the execution of the following code on a CDC6600:

$$x_7 \leftarrow x_1/x_7$$

```
X1 <- X2*X3
```

```
X4 <- X5*X6
```

```
X1 <- X1+X4
```

```
X4 <- X1*X7
```

```
X7 <- X1/X7
```

$X1 \leftarrow X2 * X3$

$X4 \leftarrow X5 * X6$

```
X1 <- X1+X4
```

```
X4 <- X1*X7
```

-2	Missed conflict