16.482 / 16.561: Computer Architecture and Design

Summer 2014

Midterm Exam
June 5, 2014

Name: ____ **ID #:** _________________________

For this exam, you may use a calculator and two 8.5” x 11” double-sided page of notes. All other electronic devices (e.g., cellular phones, laptops) are prohibited. If you have a cellular phone, please turn it off prior to the start of the exam to avoid distracting other students.

The exam contains 6 questions for a total of 100 points. Please answer the questions in the spaces provided. If you need additional space, use the back of the page on which the question is written and clearly indicate that you have done so.

You will be provided with two pages (1 double-sided sheet) that contain a list of the MIPS instructions we have covered thus far. You do not have to submit these pages when you turn in your exam.

You will have three hours to complete this exam.

Q1: Evaluating instructions	/ 16
Q2: Binary multiplication	/ 14
Q3: IEEE floating-point format	/ 20
Q4: Pipelining	/ 14
Q5: Dynamic branch prediction	/ 20
Q6: Dependences	/ 16
TOTAL SCORE	/ 100

1. (16 points) ***Evaluating instructions***

For each part of the following question, assume the following initial state. Note that your answers to each part should use the values below—your answer to part (a), for example, should not affect your answer to part (b).

- `$s1 = 0x00000005, $s2 = 0x00000003, $s3 = 0x00002000`
- Contents of memory (all values are in hexadecimal)

Address				
0x00002014	BE	12	EF	33
0x00002018	06	05	20	14

For each instruction sequence below, list **all** changed registers and/or memory locations and their new values. When listing memory values, list the entire word—for example, if a byte is written to 0x00002014, show the values at addresses 0x00002014-0x00002017.

a.
```
sub     $t0, $s1, $s2
xori    $t1, $t0, 0xFFFF
sh      $t1, 0x16($s3)
sll     $t2, $t1, 8
```

b.
```
addi    $t3, $zero, 0x2014
lh      $t4, 6($t3)
and     $t5, $t4, $s1
slt     $t6, $t5, $s2
```

2. (14 points) ***Binary multiplication***

You are given A = 6 and B = -3. Assume each operand uses four bits. Show how the binary multiplication of A * B would proceed using Booth's Algorithm.

3. (20 points) ***IEEE floating-point format***

Add the two IEEE single-precision floating-point values 0x41380000 and 0xc0980000. For full credit, you must show all work, including:

- Convert the two values into binary
- Perform the addition in binary, not decimal
- Re-encode the result in IEEE single-precision format

4. (14 points) ***Pipelining***

Consider the following code sequence for both parts of this question. Assume the use of a five-stage pipeline.

```
lw   $s0, 0($t1)
lw   $s1, 4($t1)
add  $t0, $t3, $s0
sub  $t1, $s1, $s0
add  $t2, $t1, $t0
lw   $s2, 0($t2)
add  $t3, $t1, $t2
xor  $t4, $t3, $s1
sw   $s2, 4($t2)
```

For both parts of this problem, show all work for full credit.

a. (8 points) If we assume we have a five stage pipelined datapath **without forwarding**, how many cycles will the instructions above take?

4 (continued)

Again, consider the following code sequence:

```
lw   $s0, 0($t1)
lw   $s1, 4($t1)
add  $t0, $t3, $s0
sub  $t1, $s1, $s0
add  $t2, $t1, $t0
lw   $s2, 0($t2)
add  $t3, $t1, $t2
xor  $t4, $t3, $s1
sw   $s2, 4($t2)
```

b. (6 points) If we now assume a five stage pipelined datapath **with forwarding**, how many cycles will the instructions above take?

5. (20 points) ***Dynamic branch prediction***

a. (14 points) Say you are executing a program that contains two branches, as shown below. You are given the addresses of each branch in both decimal and hexadecimal.

```
      Address
Decimal   Hex
     10   0x0A  loop  …
                      …
     56   0x38        BEQ R4, R0, else
                      …
     72   0x48        BNE R7, R8, loop
```

Your processor contains an eight entry, 2-bit branch history table. Initially, entries 0-3 (the first four lines of the table) all have the state 01, and entries 4-7 (the last four lines of the table) all have the state 10.

Complete the table below to show which BHT entry is used to predict each branch, what predictions are made based on that entry, and how the state of each BHT entry changes throughout the program. You are given the actual outcome for each branch.

Loop Iteration	Branch	BHT Entry #	BHT Entry State	Pred.	Actual Outcome	New BHT Entry State
1	BEQ				T	
1	BNE				T	
2	BEQ				NT	
2	BNE				T	
3	BEQ				T	
3	BNE				T	
4	BEQ				NT	
4	BNE				NT	

5 (continued)

b. (6 points) Assume you have a (3,2) correlating predictor in the state shown below:

00	00	00	00	11	01	10	10
10	01	01	10	10	01	01	01
11	11	10	00	01	10	10	10
01	01	11	11	00	01	00	11

1	0	1

If we have a branch at address 28 (0x1C in hex), what entry of the predictor will we access, and what will the prediction be? As part of your answer, circle the appropriate entry above, and briefly explain how we determine which entry to access.

6. (16 points) ***Dependences***

Answer the following questions about the code sequence below:

```
I0:       L.D       F0, 0(R4)
I1:       ADD.D     F4, F0, F2
I2:       S.D       F4, 8(R4)
I3:       DIV.D     F6, F0, F6
I4:       MUL.D     F2, F4, F0
I5:       ADDI      R4, R4, 32
I6:       SUB.D     F4, F6, F2
I7:       S.D       F4, -16(R4)
I8:       BLT       R4, R3, I0
```

a. (8 points) List all true data dependences in this code. Assume the branch at the end of the loop is taken at least once. List your dependences in the form:

```
<register number>:<producing inst.> → <consuming inst.>
```

For example, a dependence involving R1 between "I2" and "I3" would be listed as:

```
R1: I2→I3
```

Your list should only contain true dependences—do not list any name dependences.

6 (continued)
Again, consider the following code, and assume the branch at the end of the loop is taken at least once:

```
I0:       L.D       F0, 0(R4)
I1:       ADD.D     F4, F0, F2
I2:       S.D       F4, 8(R4)
I3:       DIV.D     F6, F0, F6
I4:       MUL.D     F2, F4, F0
I5:       ADDI      R4, R4, 32
I6:       SUB.D     F4, F6, F2
I7:       S.D       F4, -16(R4)
I8:       BLT       R4, R3, I0
```

b. (4 points) List all anti-dependences in this code.

c. (4 points) List all output dependences in this code.