Ryerson University
Department of Electrical and Computer Engineering

COE 608 – Computer Organization and Architecture

Midterm Exam **June 07, 2018**

Name: [redacted]

Time limit: 1 hour 50 minutes **Examiners:** P. Siddavaatam

Notes:

a) Closed book.
b) No calculators.
c) Answer all questions **in the space provided**.
d) Circle your professor's name and hand in these sheets.
e) One double sided cheat sheet permitted.
f) **No questions** during the exam. State your assumptions.

Q1	11.8 /12
Q2	8 /12
Q3	0 /12
Q4	12 /12
Q5	5 /12
Total	36.8 /60

Q1)

a. State the five major components of a computer **2 Marks**

b. Assemble the following MIPS instruction into its binary machine representation: **10 Marks**

xori $15, $0, 0x8000

addi $21, $22, -50

a) Input, output, Memory, Control, Datapath (where?) 1.8

b) 001110 00000 01111 1000 0000 0000 0000 (5)
xori $0 $15 8 0 0 0

001000 10110 10101 1111 1111 1100 1110
addi $22 $21 -50
(5)

Q2)

a. Which instruction has the same representation as 35_{ten} ? **2 Marks**

a1. subu $s0, $s0, $s0
a2. lw $0,0($0)
a3. addi $0, $0,35
a4. subu $0, $0, $0

100011
2 3
I + ev

b. Write a series of instructions to MIPS instructions to perform an unsigned (i) 64-bit subtraction and (ii) signed 64-bit SLT. The operands are passed in registers a1: a0 and a3: a2 with the MSW in higher numbered register. The result should be returned to (i) v1: v0 and (ii) t0 with the same convention. Explain all the steps with comments using # symbol.

10 Marks

b) i)
```
lw $a1,0($a1)      # Load MSW1 into $a1
lw $a0,0($a0)      # Load LSW1 into $a0
lw $a3,0($a3)      # Load MSW2 into $a3
lw $a2,0($a2)      # Load LSW2 into $a2

subu $v1, $a1, $a3 # v1 = a1 - a3 (MSW1 - MSW2)
                   # Assuming a1 - a3 is alright
subu $v0, $a0, $a2 # v0 = a0 - a2
                   # Assuming a0 - a2 is alright
```

(3)

ii)
```
lw $a1, 0($a1)     # load MSW into $a1
lw $a0, 0($a0)     # load LSW into $a0
lw $a3, 0($a3)     # Load MSW into $a3
lw $a2, 0($a2)     # loadLSW into $a2
slt $t0, $a1, $a3  # $a1 < $a3 ? Assuming we want this
slt $t0, $a0, $a2  # $a0 < $a2 ? Assuming we want this.
```

(5)

Q3) Translate the following snippet of C code to MIPS. Use the traditional MIPS conventions for argument passing, return values, and adjusting the stack pointer.

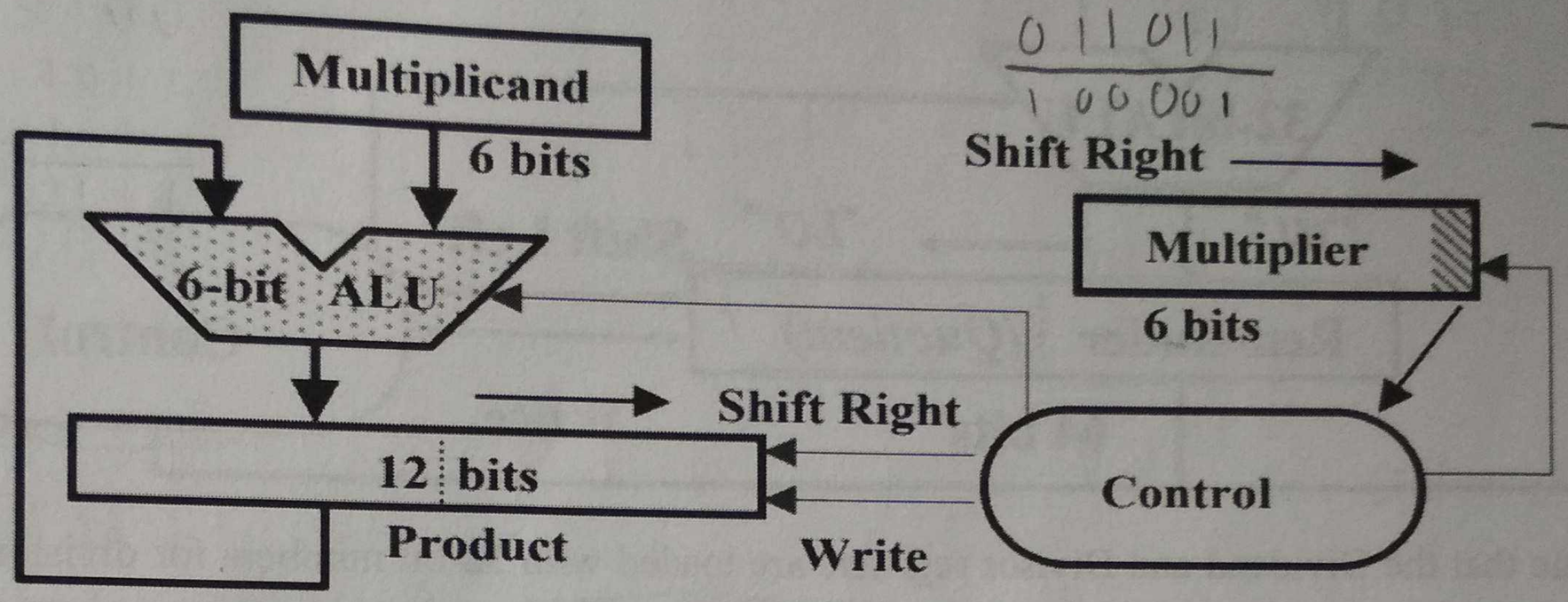
12 Marks

```
int saturate(int sum) {
if (byte_overflow(sum)) return 0xff;
else return sum;
}

int byte_overflow(int num) {
if (num >= 0x0100) return 1;
else return 0;
}
```

```
lw    $s0 , 0($s0)      # Load sum into $s0
lw    $t0 , 0           # Load $t0 = 0
add   $s2 , $0 , $0     # Initialize $s2 with 0
addi  $s2 , $s2 , 4     # add $s2 with 4
slt   $t1 , $s2 , $s0   # $s2 < $s0
bnea  $t1 , $t0 label1  # check if 4 < num
      lw  $t3 , 0($s0)  # Load sum if byte-overflow = 0
label1: addi , $t3 , 255  # Add and display FF=255
```

Q4) The block diagram of a 32-bit multiplier with optimal size ALU (adder) and registers is given below.

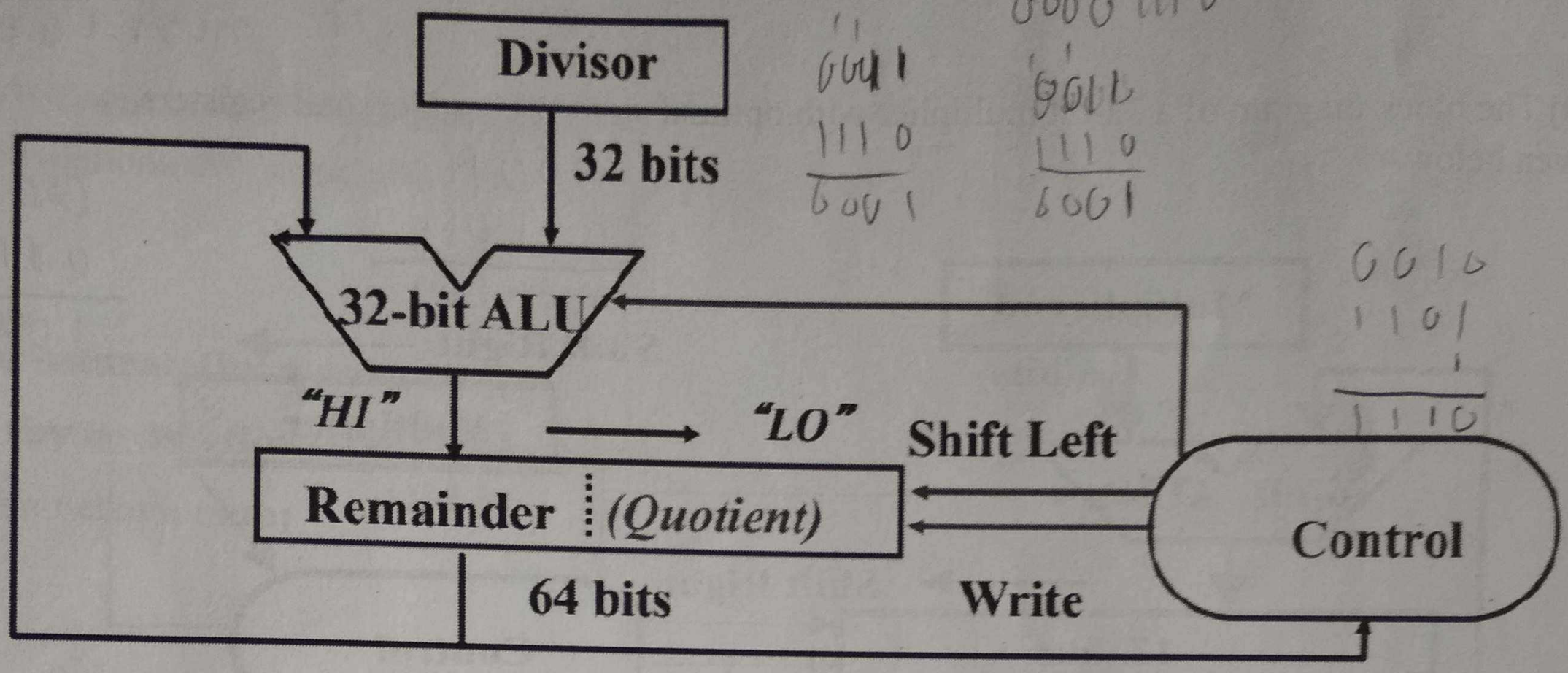


Assume that the Multiplicand and Multiplier registers are loaded with 32-bit numbers for multiplication and the product register is cleared initially as shown below. Two 32-bit unsigned binary numbers are to be multiplied using shift or add/shift operations. During this process, show the contents of all these registers that change. Determine the contents of these registers after an add/shift or just shift operation and fill up the following table after each operation.

12 Marks

Operation Shift and/or Add	**Product**	**Multiplicand**	**Multiplier**
Initial Values	000000000000	011011	010101
Add to left hand	011011000000	011011	010101
Shift Product	001101100000	011011	010101
Shift Multiplier	001101100000	011011	001010
Shift Product	000110110000	011011	001010
Shift Multiplier	000110110000	011011	000101
Add to left hand	100011110000	011011	000101
Shift Product	010001111000	011011	000101
Shift Multiplier	010001111000	011011	000010
Shift Product	001000111100	011011	000010
Shift Multiplier	001000111100	011011	000001
Add to left hand	100011011100	011011	000001
Shift Product	010001101110	011011	000001
Shift Multiplier	010001101110	011011	000000
Shift Product	001000110111	011011	000000
Shift Multiplier	001000110111	011011	000000

Q5) The block diagram of a 32-bit divider (version 3) with optimal size ALU (adder) and registers is given below.



Assume that the Dividend and Divisor registers are loaded with 32-bit numbers for division and the remainder register is set initially as shown below. Two 32-bit unsigned binary numbers are to be divided using shift or sub/shift operations. During this process, show the contents of all these registers that change. Determine the contents of these registers after an sub/shift or just shift operation and fill up the following table after each operation.

12 Marks

Step	Remainder	Quotient	Divisor	Rem-Div
Initial Values	0000 0111	0000	0010	
Sub/Shift	1110 1110	0000	0010	1110
Shift	0001 1100 ✓	0001	0010	1110 ?
Sub/Shift	0001 1000 ✗	0001 ✗	0010	0001
Shift	0011 0001 ✓	quotient ?	0010	0001 ?
Sub/Shift	0001 0010 ✗		0010	0001
Shift	0010 0100 ✗		0010	
	Steps are not clear		0010	
Sub/Shift			0010	
			0010	
			0010	

3
2 √7
-6
1

1
0010 √00000111
+0110
1110

10
01
1
10

11
10 √0111
-11

0011
1101
1
0001

1101
1
1110

0011
1110
0001

0100

100
10 √1000
10
000
-00
00

0000
-0010
0000
1101
1
1110

0010
1101
1
1110

1
10 √0111
-10
111

11
010 √0111
-010
011
010
011

11
010 √0111
010
0011
010
001

0110
1110
0100

$$
\begin{array}{r}
0.0111 \\
0101\overline{)0000\ 0111} \\
-0\ 0000 \\
\hline
0\ 0111 \\
-0\ 0101 \\
\hline
0\ 0010 \\
-0\ 0000 \\
\hline
0\ 0101 \\
-0\ 0101 \\
\hline
0000
\end{array}
$$