

**EE234**

**Microprocessor Systems**

**Midterm Exam 1**

**Oct. 10, 2022. (2:10pm – 3pm)**

**Instructor: Dae Hyun Kim ([daehyun@eecs.wsu.edu](mailto:daehyun@eecs.wsu.edu))**

**Name:**

**WSU ID:**

Problem	Points	
1	10	
2	10	
3	20	
4	30	
5	30	
Total	100	

## Problem #1 (Bit manipulation, 10 points)

Suppose R# is a 16-bit register. The data stored in R# is treated as an unsigned binary number. R1 has an input data. The following two instructions perform an arithmetic operation. Explain what it does (i.e., briefly explain the meaning of the data stored in R2 in terms of arithmetic operations) or draw a graph of (R1 vs. R2). Here, “arithmetic” means something like addition, subtraction, multiplication, division (quotient), division (remainder), square root, transcendental functions, etc. Ignore overflow/underflow exceptions in the operations.

```
MOV R2, R1, LSL #2
```

```
EOR R2, R2, #0x0002
```

Input:  $x_{15}x_{14} \dots x_2x_1x_0$

Output:  $x_{13}x_{12} \dots x_210$

$$Y = 4X + 2$$

## Problem #2 (Bit manipulation, 10 points)

Suppose R# is an 8-bit register.  $R_1$  and  $R_2$  are given as follows:

$$R_1 = x_7x_6x_5x_4 x_3x_2x_1x_0$$

$$R_2 = y_7y_6y_5y_4 y_3y_2y_1y_0$$

Write an assembly code to generate  $R_3$  from  $R_1$  and  $R_2$ . You can use the following instructions only. (&: logical AND, |: logical OR, ^: logical XOR)

- AND, ORR, EOR, MOV (including LSL, LSR)

$$R_3 = (1)(0)(\overline{x_1})(\overline{y_0}) (x_7\&y_7)(0)(1)(0)$$

(i.e., if  $R_3 = a_7a_6 \dots a_0$ , then  $a_7 = 1$ ,  $a_6 = 0$ ,  $a_5 = \overline{x_1}$ ,  $a_4 = \overline{y_0}$ ,  $a_3 = x_7\&y_7$ ,  $a_2 = 0$ ,  $a_1 = 1$ ,  $a_0 = 0$ .)

MOV R4, R1, LSL #4 // R4 =  $x_3x_2x_1x_0$  0000

AND R4, R4, #0x20 // R4 = 00 $x_1$ 0 0000

MOV R5, R2, LSL #4 // R5 =  $y_3y_2y_1y_0$  0000

AND R5, R5, #0x10 // R5 = 000 $y_0$  0000

ORR R4, R4, R5 // R4 = 00 $x_1y_0$  0000

EOR R4, R4, #0xB0 // R4 = 10 $\overline{x_1}\overline{y_0}$  0000

MOV R5, R1, LSR #4 // R5 = 0000  $x_7x_6x_5x_4$

MOV R6, R2, LSR #4 // R6 = 0000  $y_7y_6y_5y_4$

AND R5, R5, R6 // R5 = 0000  $(x_7\&y_7)XXX$

AND R5, R5, #0x08 // R5 = 0000  $(x_7\&y_7)000$

EOR R5, R5, #0x02 // R5 = 0000  $(x_7\&y_7)010$

ORR R3, R4, R5 // R3 = 10 $\overline{x_1}\overline{y_0}$   $(x_7\&y_7)010$

### Problem #3 (ARM assembly, 20 points)

```
main:
    MOV R1, #1
    MOV R2, #1
    MOV R3, #0
    MOV R4, #4
loop1:
    MOV R0, R3, LSL #2
    MOV R5, R2, LSL #1
    ADD R0, R0, R5
    ADD R0, R0, R1
    CMP R4, #7
    BGE end
    MOV R3, R2
    MOV R2, R1
    MOV R1, R0
    ADD R4, R4, #1
    B loop1
end:
    // end of code
```

(20 points) What is the value of the data stored in R0 when the following program ends?

Iter	R0	R1	R2	R3	R4	R5
	?	1	1	0	4	?
1	$0+2+1=3$	3	1	1	5	2
2	$4+2+3=9$	9	3	1	6	2
3	$4+6+9=19$	19	9	3	7	6
4	$12+18+19=49$					18

Thus, R0 has 49.

## Problem #4 (ARM assembly, 30 points)

What is the value of the data stored in R5 when the program ends?

```
main:
    MOV R5, #0
    MOV R1, #1
loop1:
    CMP R1, #5
    BGT end1
    MOV R2, R1
loop2:
    CMP R2, #5
    BGT end2
    MOV R3, R2
loop3:
    CMP R3, #5
    BGT end3
    ADD R5, R5, R3
    ADD R3, R3, #1
    B loop3
end3:
    ADD R2, R2, #1
    B loop2
end2:
    ADD R1, R1, #1
    B loop1
end1:
    // end of code
```

(Hint: This code has three “for” loops. You can translate the code into a C code, and then analyze it.)

```
R5 = 0
for ( R1 = 1 ; R1 <= 5 ; R1++ ) {
    for ( R2 = R1 ; R2 <= 5 ; R2++ ) {
        for ( R3 = R2 ; R3 <= 5 ; R3++ ) {
            R5 += R3;
        }
    }
}
```

R1: 1 to 5. For each R1, R2: R1 to 5. For each R2, R3: R2 to 5.

R1 = 1: R2 = 1~5: R3 = sum(1~5) + sum(2~5) + ... + sum(5~5)

R1 = 2: R2 = 2~5: R3 = sum(2~5) + sum(3~5) + ... + sum(5~5)

...

$$R1 = 5: R2 = 5: R3 = \text{sum}(5\sim 5)$$

$$\text{Thus, } R5 \text{ has } 1*\text{sum}(1\sim 5) + 2*\text{sum}(2\sim 5) + 3*\text{sum}(3\sim 5) + \dots + 5*\text{sum}(5\sim 5) = 15 + 2*14 + 3*12 + 4*9 + 5*5 = 15 + 28 + 36 + 36 + 25 = 140$$

## Problem #5 (ARM assembly, 30 points)

Translate the following C code into an assembly code.

```
int a, b, c;

a = 0;
b = 0;
c = 0;
while ( (a < 10) && (b < 10) ) {
    a = a + 1;
    b = b + a;
    if ( c == 20 )
        break; // get out of the while loop
    c = c + b;
}
```

- Use the assembly instructions listed in the last page only.
- a is in R0, b is in R1, and c is in R2.
- The exit point (the end of the if statement) could be just an address label.

```
MOV R0, #0
MOV R1, #0
MOV R2, #0
```

while:

```
CMP R0, #10
BGE end
CMP R1, #10
BGE end
ADD R0, R0, #1
ADD R1, R1, R0
CMP R2, #20
BNE end
ADD R2, R2, R1
B while
```

end: