EE234

Microprocessor Systems

Midterm Exam

Oct. 14, 2020. (2:10pm – 3pm)

Instructor: Dae Hyun Kim (daehyun@eecs.wsu.edu)

Name:

WSU ID:

Problem	Points	
1	10	
2	10	
3	10	
4	10	
5	10	
6	10	
7	20	
Total	80	

Problem #1 (Bit manipulation, 10 points)

Suppose R# is an 8-bit register. The data stored in R# is treated as an unsigned binary number. The following instruction performs 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.

EOR R2, R1, #0x03

Input: $x_7x_6x_5x_4x_3x_2x_1x_0$

Output: $x_7x_6x_5x_4x_3x_2\overline{x_1x_0}$

Answer: Suppose X and Y are the values in R1 and R2, respectively.

If $X = 4n$ (where n is an integer), $Y = 4n + 3$.

If $X = 4n + 1$, $Y = 4n + 2$.

If $X = 4n + 2$, $Y = 4n + 1$.

If $X = 4n + 3$, $Y = 4n$.

Well, if you want to combine all the cases into one formula, it could be

$$Y = X + (3 - (X\%4))$$

where % is the MOD operation.

Problem #2 (Bit manipulation, 10 points)

Suppose R# is an 8-bit register. The data stored in R# is treated as an unsigned binary number. The following instruction performs 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.

OR R2, R1, #0x80

Input: $x_7x_6x_5x_4x_3x_2x_1x_0$

Output: $1x_6x_5x_4x_3x_2x_1x_0$

Answer: Suppose X and Y are the values in R1 and R2, respectively.

If $X < 128$ (i.e., $x_7 = 0$), $Y = 128 + X$.

If $X \geq 128$ (i.e., $x_7 = 1$), $Y = X$.

Problem #3 (Bit manipulation, 10 points)

Suppose R# is an 8-bit register. R1 ($x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$) has an input data. We want to generate the following signal from R1.

$$\text{R2} = (x_7\ \overline{x_6}\ 1\ 0\ x_3\ \overline{x_2}\ 1\ 0)$$

Use the instructions in the instruction sheet to generate R2. Try to minimize # instructions you use.

EOR R2, R1, #0x44 // $x_7\ \overline{x_6}\ x_5\ x_4\ x_3\ \overline{x_2}\ x_1\ x_0$

AND R2, R2, #0xEE // $x_7\ \overline{x_6}\ x_5\ 0\ x_3\ \overline{x_2}\ x_1\ 0$

OR R2, R2, #0x22 // $x_7\ \overline{x_6}\ 1\ 0\ x_3\ \overline{x_2}\ 1\ 0$

Problem #4 (Bit manipulation, 10 points)

Suppose R# is an 8-bit register. R1 ($x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$) has an input data. We want to generate the following signal from R1.

$$\text{R2} = (x_3\ x_2\ x_1\ x_0\ x_7\ x_6\ x_5\ x_4)$$

Use the instructions in the instruction sheet to generate R2. Use only R1 and R2 (i.e., don't use any other registers.) You don't need to preserve the input data. Try to minimize # instructions you use. (≤3 instructions: 10 points, 4 instructions: 5 points, ≥5 instructions: 2 points)

LSL R2, R1, #4 // R1 ($x_7\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$), R2 ($x_3\ x_2\ x_1\ x_0\ 0\ 0\ 0\ 0$)

LSR R1, R1, #4 // R1 ($0\ 0\ 0\ 0\ x_7\ x_6\ x_5\ x_4$), R2 ($x_3\ x_2\ x_1\ x_0\ 0\ 0\ 0\ 0$)

OR R2, R1, R2 // R2 ($x_3\ x_2\ x_1\ x_0\ x_7\ x_6\ x_5\ x_4$)

Problem #5 (ARM assembly, 10 points)

What is the value of the data stored in R2 when the program ends?

```
  MOV R1, #0
  MOV R2, #0
loop1:
  ADD R1, R1, #1
  MOV R3, R1
  AND R4, R3, #0x01
  LSR R3, R3, #1
  AND R4, R4, R3
  LSR R3, R3, #1
  AND R4, R4, R3
  ADD R2, R2, R4
  CMP R1, #255
  BLT loop1
end:
    // end of code
```

R1: $x_7\ x_6\ \dots\ x_0$.

MOV R3, R1: R3 = $x_7\ x_6\ \dots\ x_0$

AND R4, R3, #0x01: R4 = $0\ 0\ 0\ 0\ 0\ 0\ 0\ x_0$

LSR R3, R3, #1: R3 = $0\ x_7\ \dots\ x_1$

AND R4, R4, R3: R4 = $0\ 0\ 0\ 0\ 0\ 0\ 0\ (x_0 \& x_1)$

LSR R3, R3, #1: R3 = $0\ 0\ x_7\ \dots\ x_2$

AND R4, R4, R3: R4 = $0\ 0\ 0\ 0\ 0\ 0\ 0\ (x_0 \& x_1 \& x_2)$

ADD R2, R2, R4: R2 is increased by 1 whenever $x_0 = 1$, $x_1 = 1$, $x_2 = 1$.

How many times? 32 (from $x_7 \dots x_3 = 00000$ to $x_7 \dots x_3 = 11111$).

Thus, R2 will have **32**.

This code counts # integers in the form of $8n + 7$ between 1 and 255.

Problem #6 (ARM assembly, 10 points)

What is the value of the data stored in R2 when the program ends?

```
  MOV R1, #0
  MOV R2, #1
  MOV R3, #2
loop1:
  ADD R4, R1, R2
  ADD R3, R3, #1
  MOV R1, R2
  MOV R2, R4
  CMP R3, #10
  BNE loop1
end:
    // end of code
```

R4 = 1, R3 = 3, R1 = 1, R2 = 1, R3 != 10

R4 = 2, R3 = 4, R1 = 1, R2 = 2, R3 != 10

R4 = 3, R3 = 5, R1 = 2, R2 = 3, R3 != 10

R4 = 5, R3 = 6, R1 = 3, R2 = 5, R3 != 10

R4 = 8, R3 = 7, R1 = 5, R2 = 8, R3 != 10

R4 = 13, R3 = 8, R1 = 8, R2 = 13, R3 != 10

R4 = 21, R3 = 9, R1 = 13, R2 = 21, R3 != 10

R4 = 34, R3 = 10, R1 = 21, R2 = 34, R3 == 10

Thus, R2 has **34**.

This code calculates the 10th element of the Fibonacci sequence ($X_{n+2} = X_{n+1} + X_n$ with $X_1 = 0$ and $X_2 = 1$).

Problem #7 (ARM assembly, 20 points)

Let's use the 32-bit ARM architecture, i.e., R# is a 32-bit register and the register file has 16 registers (you can use R0~R12 only). R0 has a positive number (given to you). We want to check whether the number in R0 is an integer multiple of 3 (i.e., $3n$) or not. If it is, we set R1 to 1. If not, we set R1 to 0. Here is an algorithm for that.

1) If R0 is 0, R1 = 1. Done.

2) If R0 is 1, R1 = 0. Done.

3) If R0 is 2, R1 = 0. Done.

4) If not (i.e., R0 $\geq$ 3), subtract 3 from R0 (i.e., R0 = R0 – 3).

5) Go back to 1).

Write an assembly code running the above algorithm. Use only the instructions shown in the instruction sheet (but do not use LDR and STR). The performance of the code doesn't matter as long as the code works.

```
  MOV R1, #1
loop:
  CMP R0, #0
  BEQ end
  CMP R0, #1
  BEQ case 2
  CMP R0, #2
  BEQ case 2
  SUB R0, R0, #3
  B loop
case2:
  MOV R1, #0
end:
  // end of code
```