EE234

Microprocessor Systems

Midterm Exam

Nov. 13, 2020. (2:10pm – 3pm)

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

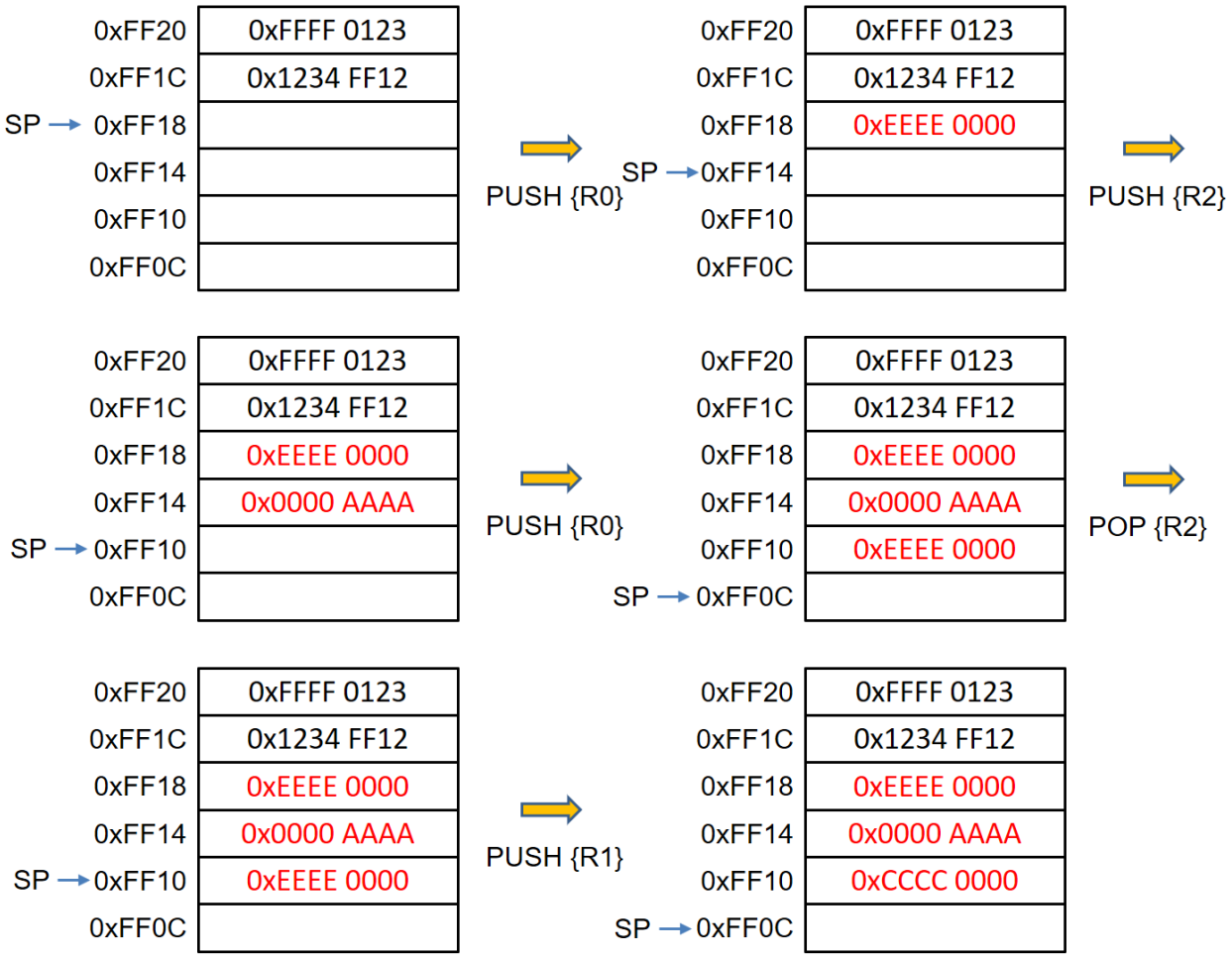
Name:

WSU ID:

Problem	Points	
1	10	
2	20	
3	30	
4	30	
Total	90	

Problem #1 (Stack, 10 points)

R0 has 0xEEEE 0000, R1 has 0xCCCC 0000, and R2 has 0x0000 AAAA. Show the contents of the stack memory and the stack pointer after the PUSH or POP operations below.



Problem #2 (Stack, 20 points)

Answer the following questions for the assembly code shown below.

```
main:
  MOV R0, #5
  MOV R2, #0
  BL run
  B finish
run:
  PUSH {R14}
  PUSH {R0}
  CMP R0, #1
  BEQ run_1
  SUB R0, R0, #1
  BL run
  ADD R0, R0, #1
  ADD R2, R2, R0
run_2:
  POP {R0}
  POP {R14}
  BX LR
run_1:
  MOV R2, #1
  B run_2
finish:
.end
```

(1) (10 points) What is the value stored in R2 when the program ends?

(2) (20 points) How many times is the “PUSH {R14}” statement executed?

Let’s assume that the memory address of the instruction “B finish” is A1 and that of “ADD R0, R0, #1” is A2.

R0 = 5, R2 = 0. Call run.

PUSH: Stack {A1, 5}

R0 = 5 – 1 = 4

PUSH: Stack {A1, 5, A2, 4}

R0 = 4 – 1 = 3

PUSH: Stack {A1, 5, A2, 4, A2, 3}

R0 = 3 – 1 = 2

PUSH: Stack {A1, 5, A2, 4, A2, 3, A2, 2}

R0 = 2 – 1 = 1

PUSH: Stack {A1, 5, A2, 4, A2, 3, A2, 2, A2, 1}

run_1: R2 = 1

POP: Stack {A1, 5, A2, 4, A2, 3, A2, 2}, R0 = 1, R14 = A2

R0 = 1 + 1 = 2, R2 = 1 + 2 = 3

POP: Stack {A1, 5, A2, 4, A2, 3}, R0 = 2, R14 = A2

R0 = 2 + 1 = 3, R2 = 3 + 3 = 6

POP: Stack {A1, 5, A2, 4}, R0 = 3, R14 = A2

R0 = 3 + 1 = 4, R2 = 6 + 4 = 10

POP: Stack {A1, 5}, R0 = 4, R14 = A2

R0 = 4 + 1 = 5, R2 = 10 + 5 = 15

POP: Stack {}, R0 = 5, R14 = A1

Done

(1) R2 has 15 (which actually is the result of 1 + 2 + 3 + 4 + 5).

(2) PUSH {R14} is executed five times.

Problem #3 (Stack, 30 points)

Answer the following questions for the assembly code shown below.

```
main:
  MOV R0, #5
  MOV R2, #0
  BL run
  B finish
run:
  PUSH {R14}
  PUSH {R1}
  PUSH {R0}
  CMP R0, #1
  BEQ run_1
  CMP R0, #2
  BEQ run_1
  SUB R0, R0, #1
  BL run
  MOV R1, #0
  ADD R1, R1, R2
  SUB R0, R0, #1
  BL run
  ADD R1, R1, R2
  MOV R2, R1
run_2:
  POP {R0}
  POP {R1}
  POP {R14}
  BX LR
run_1:
  MOV R2, #1
  B run_2
finish:
.end
```

(1) (10 points) What is the value stored in R2 when the program ends?

(2) (20 points) How many times is the “PUSH {R14}” statement executed?

Let’s assume that the memory address of the instruction “B finish” is A1, that of “MOV R1, #0” is A2, and that of the second “ADD R1, R1, R2” instruction is A3.

R0 = 5, R2 = 0. Call run.

PUSH: Stack {A1, X, 5}

SUB 1st: R0 = 4

PUSH: Stack {A1, X, 5, A2, X, 4}

SUB 1st: R0 = 3

PUSH: Stack {A1, X, 5, A2, X, 4, A2, X, 3}

SUB 1st: R0 = 2

PUSH: Stack {A1, X, 5, A2, X, 4, A2, X, 3, A2, X, 2}

run_1: R2 = 1.

POP: Stack {A1, X, 5, A2, X, 4, A2, X, 3}, R0 = 2, LR = A2

R1 = 0, R1 = 0 + 1 = 1

SUB 2nd: R0 = 1

PUSH: Stack {A1, X, 5, A2, X, 4, A2, X, 3, A3, 1, 1}

run_1: R2 = 1.

POP: Stack {A1, X, 5, A2, X, 4, A2, X, 3}, R0 = 1, R1 = 1, LR = A3

R1 = 1 + 1 = 2, R2 = 2

POP: Stack {A1, X, 5, A2, X, 4}, R0 = 3, R1 = X, LR = A2

R1 = 0, R1 = 0 + 2 = 2

SUB 2nd: R0 = 2

PUSH: Stack {A1, X, 5, A2, X, 4, A3, 2, 2}

run_1: R2 = 1.

POP: Stack {A1, X, 5, A2, X, 4}, R0 = 2, R1 = 2, LR = A3

R1 = 2 + 1 = 3, R2 = 3

POP: Stack {A1, X, 5}, R0 = 4, R1 = X, LR = A2

R1 = 0, R1 = 0 + 3 = 3

R0 = 3

PUSH: Stack {A1, X, 5, A3, 3, 3}

R0 = 2

PUSH: Stack {A1, X, 5, A3, 3, 3, A2, 3, 2}

run_1: R2 = 1.

POP: Stack {A1, X, 5, A3, 3, 3}, R0 = 2, R1 = 3, LR = A2

R1 = 0, R1 = 0 + 1, R0 = 1

PUSH: Stack {A1, X, 5, A3, 3, 3, A3, 1, 1}

run_1: R2 = 1.

POP: Stack {A1, X, 5, A3, 3, 3}, R0 = 1, R1 = 1, LR = A3

R1 = 1 + 1 = 2, R2 = 2

POP: Stack {A1, X, 5}, R0 = 3, R1 = 3, LR = A3

R1 = 3 + 2 = 5, R2 = 5

POP: Stack {}, R0 = 5, R1 = X, LR = A1

Thus, when it’s done, R2 = 5. PUSH {R14} is executed nine times.

This function computes the Fibonacci sequence f(n) = f(n-1) + f(n-2) recursively with f(1) = f(2) = 1.

Problem #4 (C to Assembly, 30 points)

Write an assembly code for the following C code.

```
int main () {
  int x = 5;
  int y = find (x);
  ...
}

int find (int n) {
  if ( n == 1 )
    return 0;
  else if ( n % 2 == 0 )  // i.e., if n is an even number
    return find (n/2);
  else
    return find (3*n+1);
}
```

- Use BL and BX for the recursive function calls.
- R0 is used for the variable x.
- R1 is used for the variable y.
- For the division, use the logical shift right (LSR) instruction. For the multiplication, use the logical shift left (LSL) and addition (ADD) instructions.
- Here is my code for the main function. You implement the “find” function in assembly.

```
main:

  MOV R0, #5  // int x = 5;

  BL find

find:

  // you implement this function
```

```
main:
  MOV R0, #5
  MOV R2, R0  // just to preserve R0
  BL find
  B finish

find:  // R2: n
  CMP R2, #1
  BEQ find_ret0
  AND R3, R2, #1
  CMP R3, #0  // even
  BEQ find_even
  MOV R3, R2, LSL #1  // or just LSL R3, R2, #1
  ADD R2, R2, R3  // 3*n
  ADD R2, R2, #1  // 3n+1
  PUSH {LR}
  BL find
  POP {LR}
  BX LR

find_ret0:
  MOV R1, #0
  BX LR

find_even:
  MOV R2, R2, LSR #1  // or just LSR R2, R2, #1
  PUSH {LR}
  BL find
  POP {LR}
  BX LR
```