EE234

Microprocessor Systems

Midterm Exam 2

Nov. 16, 2022. (2:10pm – 3pm)

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

Name:

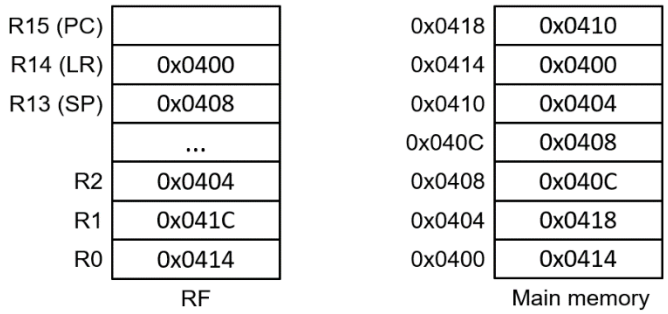
WSU ID:

Problem	Points	
1	10	
2	30	
3	40	
Total	80	

Problem #1 (Stack, 10 points)

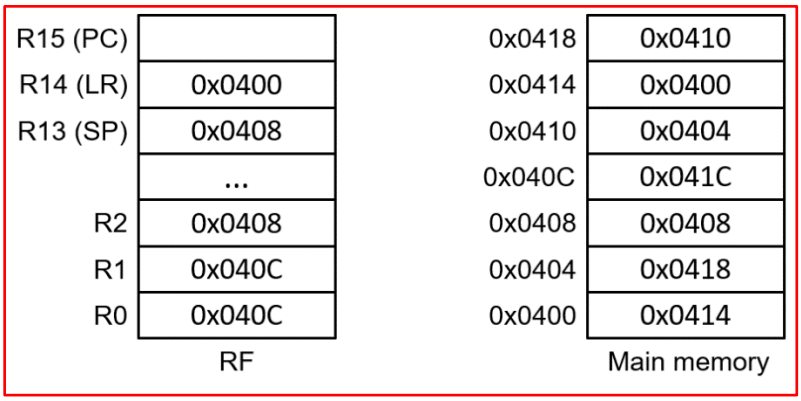
* Notice that SP (R13) points to the bottommost element in the stack.

The following shows the register file (RF) and the main memory map.



Show the values of R0, R1, R2, and the main memory map after the following instructions are executed.

```
POP {R0}
POP {R2}
PUSH {R1}
PUSH {R0}
POP {R1}
PUSH {R2}
```



Problem #2 (Stack, 30 points)

Answer the following questions for the assembly code shown below.

```
main:
  MOV R0, #4
  MOV R1, #6
  MOV R2, #0
  PUSH {R2}
  PUSH {R0}
  PUSH {R1}
  BL find
A1 POP {R1}
  POP {R0}
  POP {R2}
  B end
end:
  // end of code
```

```
find:
  PUSH {LR}
  PUSH {R0}
  PURH {R1}
  LDR R0, [SP, #16]
  LDR R1, [SP, #12]
  CMP R0, R1
  BLT find_1
  CMP R0, R1
  BGT find_2
  STR R0, [SP, #20]
find_ret:
  POP {R1}
  POP {R0}
  POP {LR}
  BX LR
```

```
find_1:
  ADD R0, R0, #4
  PUSH {R2}
  PUSH {R0}
  PUSH {R1}
  BL find
A2 POP {R2}
  POP {R2}
  POP {R2}
  STR R2, [SP, #20]
  B find_ret

find_2:
  ADD R1, R1, #6
  PUSH {R2}
  PUSH {R0}
  PUSH {R1}
  BL find
A3 POP {R2}
  POP {R2}
  POP {R2}
  STR R2, [SP, #20]
  B find_ret
```

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

(2) (10 points) How many times is the “PUSH {LR}” statement (the first line of the “find” subroutine) executed?

(3) (10 points) If we replace #4 by #7 (the first line after the “main:” label and the first line after the “find_1” label) and #6 by #8 (the second line after the “main:” label and the first line after the “find_2” label), what value will R2 have when the program ends?

R0=4, R1=6, R2=0

Stack={0, 4, 6}

BL find: Stack={0, 4, 6, A1, 4, 6}, R0=4, R1=6, (R2=0)

R0 < R1 (find_1): R0=8 (R1=6, R2=0), Stack={0, 4, 6, A1, 4, 6, 0, 8, 6}

BL find: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6}, R0=8, R1=6 (R2=0)

R0 > R1 (find_2): R1=12 (R0=8, R2=0), Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12}

BL find: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12}, R0=8, R1=12 (R2=0)

R0 < R1 (find_1): R0=12, (R1=12, R2=0), Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12, 0, 12, 12}

BL find: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12, 0, 12, 12, A2, 12, 12}, R0=12, R1=12

R0 == R1: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12, 12, 12, 12, A2, 12, 12} => 3 POPs: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12, 12, 12, 12}, LR=A2

A2: 3 POPs: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 0, 8, 12, A3, 8, 12}, R2=12, STR: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 12, 8, 12, A3, 8, 12}

find_ret: 3 POPs: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6, 12, 8, 12}, LR=A3

A3: 3 POPs: Stack={0, 4, 6, A1, 4, 6, 0, 8, 6, A2, 8, 6}, R2=12, STR: Stack={0, 4, 6, A1, 4, 6, 12, 8, 6, A2, 8, 6}

find_ret: 3 POPs: Stack={0, 4, 6, A1, 4, 6, 12, 8, 6}, LR=A2

A2: 3 POPs: Stack={0, 4, 6, A1, 4, 6}, R2=12, STR: Stack={12, 4, 6, A1, 4, 6}

find_ret: 3 POPs: Stack={12, 4, 6}, LR=A1

A1: 3 POPs: Stack={}, R0=4, R1=6, R2=12

(1) R2 = 12

(2) PUSH {LR} is executed four times.

(3) 56 (the code computes the least common multiple)

Problem #3 (Subroutines and Stack, 40 points)

The “com” function below computes f(n) recursively for the sequence f(n) = f(n-1) + f(n-2) + f(n-3) with the initial values f(1) = f(2) = f(3) = 1.

```
int main () {
  int n, s;
  ...
  s = com (n);
}
```

```
int com (int n) {
  if ( (n >= 1) && (n <= 3) )
    return 1;
  else
    return com(n-1) + com(n-2) + com(n-3);
}
```

Here is the main function:

```
main:
  // Addr(n) = SP+4
  // Addr(s) = SP
  LDR R0, [SP, #4]
  PUSH {R0}  // placeholder for the return value
  PUSH {R0}  // n
  BL com
  POP {R0}
  POP {R0}  // s
  STR R0, [SP]
  // end of code
```

Write an assembly code for the “com” function.

```
PUSH {LR}
PUSH {R0}
PUSH {R1}
PUSH {R2}
LDR R0, [SP, #16]  // n
CMP R0, #1
BLT ret_1
CMP R0, #3
BGT ret_1
// com(n-1)
SUB R1, R0, #1  // n-1
PUSH {R1}  // placeholder for the return value
PUSH {R1}  // n-1
BL com
POP {R1}
POP {R2}  // R2 = com(n-1)
```

```
    // com(n-2)
    SUB R1, R0, #2  // n-2
    PUSH {R1}  // placeholder for the return value
    PUSH {R1}  // n-2
    BL com
    POP {R1}
    POP {R1}  // R1 = com(n-2)
    ADD R2, R2, R1  // R2 = com(n-1) + com(n-2)
    // com(n-3)
    SUB R1, R0, #3  // n-3
    PUSH {R1}  // placeholder for the return value
    PUSH {R1}  // n-3
    BL com
    POP {R1}
    POP {R1}  // R1 = com(n-3)
    ADD R2, R2, R1  // R2 = com(n-1) + com(n-2) + com(n-3)
    STR R2, [SP, #20]
    B ret
ret_1:
    MOV R0, #1
    STR R0, [SP, #20]
ret:
    POP {R2}
    POP {R1}
    POP {R0}
    POP {LR}
    BX LR
```