# Homework Assignment 3

## (Due 2:00pm, Oct. 26, email to daehyun.kim@wsu.edu)

1. (20 points) Make an assembly source code for the following C code. You will have to use the subroutine-call-related instructions (BL and BX).

```
int main () {
  int a = 30;
  int b = 40;
  int c = 50;

  int d = comp(a, b, c);
  d = a + b + c + d;
}
```

```
int comp (int x, int y, int z) {
  return (x + y – z);
}
```

- The main procedure in assembly is used for the main function.
- R0 is used for int a
- R1 is used for int b
- R2 is used for int c
- R3 is used for int d
- You don’t need to use the stack in the comp subroutine.
- The following is my code for the initialization of the variable. Complete the rest of that.

```
main:
  MOV R0, #30
  MOV R1, #40
  MOV R2, #50
  BL comp  // the return value will be placed in R3
  ADD R3, R3, R0
  ADD R3, R3, R1
  ADD R3, R3, R2
  B finish

comp:
  ADD R3, R0, R1
  SUB R3, R3, R2
  BX LR

finish:
.end
```

2. (30 points) Make an assembly source code for the following C code. You will have to use the subroutine-call-related instructions (BL and BX) and also stack-related instructions (either PUSH/POP or manually adjust the stack pointer). Assume that R0 is used for int a and R1 is used for int b.

```
int main () {
  int a = 100;

  int b = sum(a);
}
```

```
int sum (int x) {
  if (x == 1)
    return 1;
  else
    return x+sum(x-1);
}
```

```
main:
  MOV R0, #100
  MOV R2, R0
  BL sum
  B finish

sum:
  CMP R2, #1
  BNE sum1
  MOV R1, #1
  BX LR
sum1:
  SUB R2, R2, #1  // x-1
  PUSH {R14}
  BL sum  // sum (x-1)
  ADD R2, R2, #1  // restore x
  ADD R1, R2, R1  // x + sum(x-1)
  POP {R14}
  BX LR

finish:
.end
```

3. (20 points) The following code does a certain task. If you run the program, it sometimes inserts (push) some data into the stack and sometimes removes (pop) the data from the stack. What is the maximum number of elements stored in the stack while it is running?

```
main:
  MOV R0, #1
  MOV R1, #10
  BL comp
  B finish

comp:
  CMP R0, R1
  BLT inc0
  BX LR

inc0:
  PUSH {R0}
  PUSH {R14}
  ADD R0, R0, #2
  BL inc1
  POP {R14}
  POP {R0}
  BX LR

finish:
.end
```

```
inc1:
  PUSH {R1}
  PUSH {R14}
  ADD R1, R1, #1
  CMP R0, R1
  BGE inc1_proc1
  BL inc0
inc1_proc1:
  POP {R14}
  POP {R1}
  BX LR
```

| | | |
|---|---|---|
| inc0: 2 pushes. R0 = 3. | inc1: 2 pushes. R1 = 11. | R0 < R1 |
| inc0: 2 pushes. R0 = 5. | inc1: 2 pushes. R1 = 12. | R0 < R1 |
| inc0: 2 pushes. R0 = 7. | inc1: 2 pushes. R1 = 13. | R0 < R1 |
| inc0: 2 pushes. R0 = 9. | inc1: 2 pushes. R1 = 14. | R0 < R1 |
| inc0: 2 pushes. R0 = 11. | inc1: 2 pushes. R1 = 15. | R0 < R1 |
| inc0: 2 pushes. R0 = 13. | inc1: 2 pushes. R1 = 16. | R0 < R1 |
| inc0: 2 pushes. R0 = 15. | inc1: 2 pushes. R1 = 17. | R0 < R1 |
| inc0: 2 pushes. R0 = 17. | inc1: 2 pushes. R1 = 18. | R0 < R1 |
| inc0: 2 pushes. R0 = 19. | inc1: 2 pushes. R1 = 19. | R0 ≥ R1 |

After this, it just pops until it comes back to the main procedure.

Thus, the max. # elements stored in the stack is 36.