Homework Assignment 3

(Due 2:10pm, Nov. 9, email to daehyun.kim@wsu.edu or submit a hardcopy)

You should use the following instructions only.

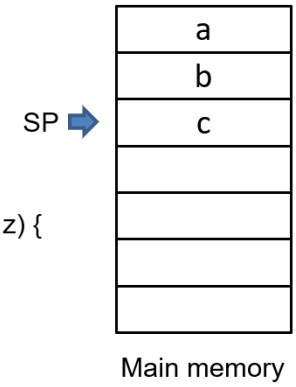
- Instructions
  - ADD, SUB
  - AND, ORR, EOR
  - CMP, BGE/BLT/BGT/BLE/BEQ/BNE
  - B, BL, BX
  - MOV
  - LDR, STR

1. (50 points) Write an assembly code for the following C code (the line c=comp() in the main function and the comp() function).

```
int main () {
  int a, b, c;
  ...
  c = comp (a,b,a+b);
  ...
}

int comp (int x, int y, int z) {
  if ( (x-y) > z )
    return 1;
  else
    return 0;
}
```

(a)



(b)

- In the main function, assume that R0-R12 are being used by other variables (right before the function call c=comp(a,b,a+b)). This means, if you want to use any of them, you should preserve their values.
- Use the stack memory for the function arguments and the return value.
- You don't need to preserve the value of LR in the comp function because it is a leaf function.

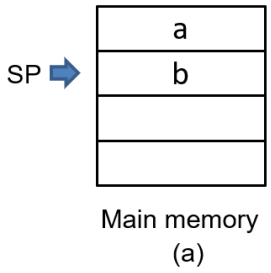
```
main:
  PUSH {R0, R1, R2}
  LDR R1, [SP, #16]  // b
  LDR R0, [SP, #20]  // a
  ADD R2, R0, R1  // a+b
  PUSH {R0}  // ret
  PUSH {R0}  // x=a
  PUSH {R1}  // y=b
  PUSH {R2}  // z=a+b
  BL comp
  LDR R0, [SP, #12]  // ret
  STR R0, [SP, #28]  // c = ret
  POP {R0}
  POP {R0}
  POP {R0}
  POP {R0}
  POP {R0, R1, R2}  // restore
```

```
comp:
  PUSH {R0, R1}
  LDR R0, [SP, #16]  // x
  LDR R1, [SP, #12]  // y
  SUB R0, R0, R1  // x-y
  LDR R1, [SP, #8]  // z
  CMP R0, R1
  BGT comp_ret_1
  MOV R0, #0
  B comp_done
comp_ret_1:
  MOV R0, #1
comp_done:
  STR R0, [SP, #20]
  POP {R0, R1}
  BX LR
```

2. (50 points) Write an assembly code for the following C code (the line b=add(a) and the add() function.

```
int main () {
  int a, b;
  ...
  b = add (a);
  ...
}
```

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



Main memory
(a)

- In the main function, assume that R0-R12 are being used by other variables (right before the function call b = add(a)). This means, if you want to use any of them, you should preserve their values.
- Use the stack memory for the function arguments and the return value.

```
main:
  PUSH {R0}
  LDR R0, [SP, #8]  // a
  PUSH {R0}  // ret
  PUSH {R0}  // x=a
  BL add
  LDR R0, [SP, #4]  // ret
  STR R0, [SP, #12]  // b = ret
  POP {R0, R0}
  POP {R0}  // restore
```

```
add:
  PUSH {R0, R1, LR}
  LDR R0, [SP, #12]  // x
  CMP R0, #1
  BNE add_not_equal
  // return 1
  MOV R0, #1
  STR R0, [SP, #16]
  POP {R0, R1, LR}
  BX LR

add_not_equal:
  SUB R1, R0, #1  // x-1
  PUSH {R0}  // ret
  PUSH {R1}  // x-1
  BL add
  LDR R1, [SP, #4]  // ret
  ADD R0, R0, R1  // x + add(x-1)
  POP {R0, R1}
  STR R0, [SP, #16]  // ret
  POP {R0, R1, LR}
  BX LR
```