Homework Assignment 4

(Due 2:10pm, Oct. 29, email to daehyun.kim@wsu.edu or submit a hardcopy)

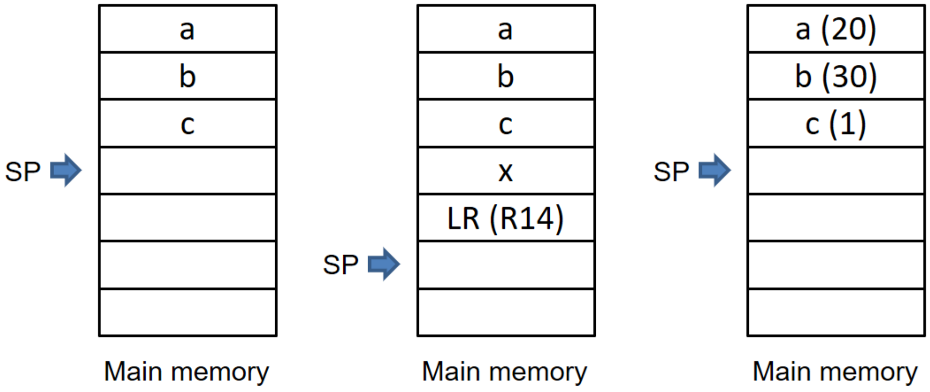
You can use the following instructions only for this homework.

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

1. (30 points) Make an assembly code for the following C code shown in (a).

```
int main () {
  int a, b, c;
  a = 20;
  b = 30;
  c = comp (a+b);
...
}

int comp (int x) {
  if ( x > 0 )
    return 1;
  else
    return 0;
}
```



(a) (b) (c) (d)

- Assume that (b) shows the memory map when the main function begins.
- (c) shows the memory map at the time the "comp" function begins (i.e., a+b is passed to x through the stack).
- (d) shows the memory map after the last line "c = comp (a+b);".
- Use R12 for the return value.
- Use R8-R11 for temporary registers in the comp function.

main:

```
MOV R8, #20
STR R8, [SP, #12]     // a = 20; R8 = a
MOV R9, #30
STR R9, [SP, #8]      // b = 30, R9 = b
ADD R10, R8, R9       // R10 = a + b
```

```
        // the caller should prepare the function argument.
        PUSH {R10}
        BL comp
        ADD SP, SP, #4          // this is just adjusting the SP back to its original location. (we can use POP too, but the popped element won’t be used.)
        STR R12, [SP, #4]       // store the return value in the main memory
        ...

comp:
        PUSH {LP}               // the same as “PUSH {R14}”
        LDR R8, [SP, #8]        // R8 = x
        CMP R8, #0
        BGT ret_1
        MOV R12, #0             // the return statement consists of “placing the return value in R12” and “BX LR”. If you pushed LP into the stack, you should of course pop LP from the stack before “BX LR”.
        POP {LP}
        BX LR
ret_1:
        MOV R12, #1
        POP {LP}
        BX LR
```

2. (70 points) Make an assembly code for the following C code.

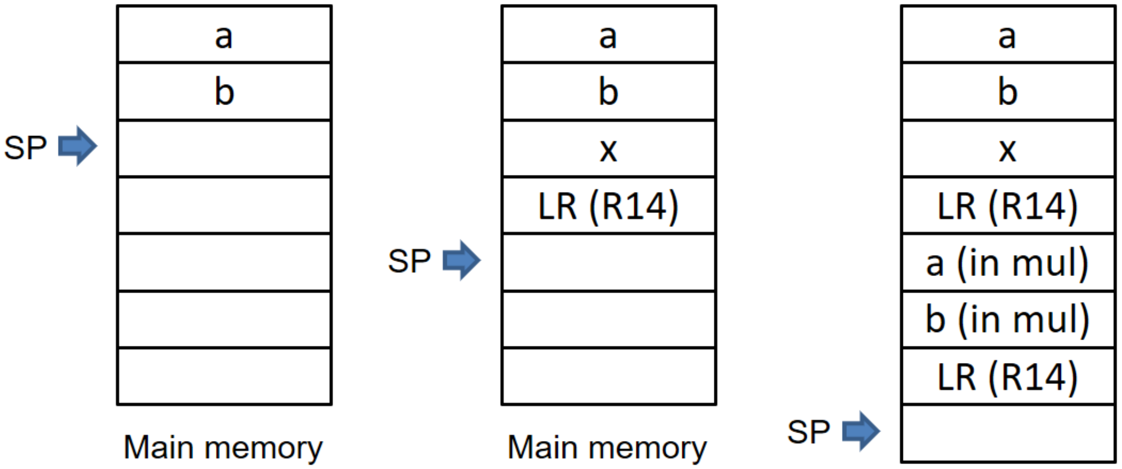
```
int main () {
  int a, b;
  a = 20;
  b = fact (a);
  ...
}
```

```
int fact (int x) {
  if ( x == 1 )
    return 1;
  else
    return mul(x, fact(x-1));
}
```

```
int mul (int a, int b) {
  int x = 0;

  for ( int i = 1 ; i <= b ; i++ )
    x = x + a;

  return x;
}
```



Main memory (a)

Main memory (b)

Main memory (c)

- Use R12 for the return values (in all the functions).
- Use (b) for the function call *fact()*.
- Use (c) for the function call *mul()*.
- Use R8-R11 for temporary registers in the functions.
- If you need more registers (R0 – R7), you should store them in the stack before you use them and then restore them later.

```
main:
        MOV R8, #20          // R8 = a
        STR R8, [SP, #8]     // a = 20
        PUSH {R8}            // x = 20
        BL fact
        ADD SP, SP, #4
        STR R12, [SP, #4]    // assign the result to b

fact:
        PUSH {LR}
        LDR R8, [SP, #8]     // R8 = x
        CMP R8, #1
        BEQ fact_1
        // if (x != 1), call x-1
        SUB R8, R8, #1       // R8 = x-1
        PUSH {R8}            // store x-1 in the stack
        BL fact              // call fact (x-1).
        POP {R8}             // just remove x-1 from the stack.
        LDR R8, [SP, #8]     // R8 = x. Notice that R12 has the return value of fact(x-1).
        // place a and b in the stack for the function call mul(a,b).
        PUSH {R8}            // a = x
        PUSH {R12}           // b = fact(x-1)
        BL mul               // when you come back from mul, R12 has mul(a,b).
        ADD SP, SP, #8       // you can also use POP twice to remove a and b from the stack.
        POP {LR}
        BX LR
fact_1:  // if ( x == 1)
        MOV R12, #1 // return 1
        POP {LR}
        BX LR

mul:
        PUSH {LR}
        LDR R10, [SP, #12]  // R10 = a
        LDR R11, [SP, #8]   // R11 = b
        MOV R8, #0          // int x = 0
        MOV R9, #1          // int i = 1
mul_comp:
        CMP R9, R11
        BGT mul_ret
        ADD R8, R8, R10     // x = x + a
        ADD R9, R9, #1      // i++
        B mul_comp
mul_ret:
        MOV R12, R8         // place the return value on R12.
        POP {LR}
        BX LR
```