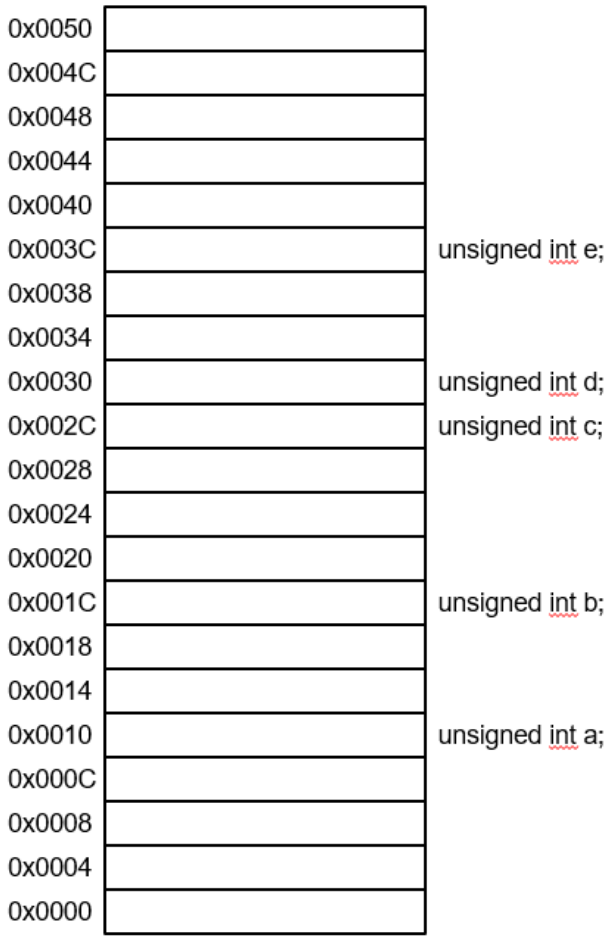
Homework Assignment 1

(Due 2:00pm, Sep. 21, email to daehyun.kim@wsu.edu)

Use the following memory map for the main memory.



1. (20 points) Make an assembly source code for the following C code. (Don't care about overflows, underflows, etc.). Do not optimize the code (for example, d is b, but don't do that.)

```
c = a + b;
d = c – a;
e = d + d;
```

LDR R0, =0x0010 // address of a

LDR R1, [R0] // R1 = a

LDR R0, =0x001C // address of b

LDR R2, [R0] // R2 = b

ADD R3, R1, R2 // R3 = a + b

```
LDR R0, =0x002C  // address of c
STR R3, [R0]  // store a+b to variable c
SUB R3, R3, R1  // R3 = c – a
LDR R0, =0x0030  // address of d
STR R3, [R0]  // store c-a to variable d
ADD R3, R3, R3  // R3 = d + d
LDR R0, =0x003C  // address of e
STR R3, [R0]  // store d+d to variable e
```

2. (20 points) Make an assembly source code for swapping the values of variables a and b. For example, if a has 10 and b has 20, a will have 20 and b will have 10 after running your code.

```
LDR R0, =0x0010  // address of a
LDR R1, [R0]  // R1 = a
LDR R0, =0x001C  // address of b
LDR R2, [R0]  // R2 = b
LDR R0, =0x0010  // address of a
STR R2, [R0]  // store the value of b to variable a
LDR R0, =0x001C  // address of b
STR R1, [R0]  // store the value of a to variable b
```

3. (30 points) We want to make an assembly source code for b=a, c=b, a=c (swapping the values of three variables, a, b, and c). For example, if a has 10, b has 20, and c has 30, a will have 30, b will have 10, and c will have 20 after running your code.

Constraint: You cannot use any other memory addresses except those for variables a, b, and c (i.e., you can access only 0x0010, 0x001C, and 0x002C).

Can you find the minimum number of registers you will need?

```
LDR R0, =0x0010  // address of a
LDR R0, [R0]  // R0 = a
LDR R1, =0x001C  // address of b
```

LDR R2, [R1] // R2 = b (currently R0=a, R1=address of b, R2 = b)

STR R0, [R1] // store the value of a to b

LDR R0, =0x002C // address of c (R0=address of c, R1=address of b, R2=b)

LDR R0, [R0] // R0 = c (R0=c, R1=address of b, R2=b)

LDR R1, =0x0010 (R0=c, R1=address of a, R2=b)

STR R0, [R1] // store the value of c to a

LDR R1, =0x002C (R0=c, R1=address of c, R2=b)

STR R2, [R1] // store the value of b to c

so, I used three registers.

More in-depth analysis:

Can we use only two registers to implement this code? In that case, the two registers will have 1) two memory addresses (e.g., the addresses of variables b and c), 2) a memory address and a value (e.g., the address of variable b and the value of c), or 3) two values (e.g., the values of b and c) at any time.

For Case 1), if we have two memory addresses in R0 and R1 (without loss of generality), we will have them because we want to load a value from one of the two memory addresses to the register file. If the loaded value is stored in a new register, it is the same as using three registers. If the loaded value is stored in R0 or R1, it is the same as Case 2.

For Case 3), if we have two values in R0 and R1 (without loss of generality), we will have them to store one of them to a certain memory address. If we use one more register to have the target memory address, it is the same as using three registers. If the target memory address is loaded to R0 or R1, it is the same as Case 2.

For Case 2), suppose we have a value (e.g., the value of b) in R0 and a memory address in R1. If we want to store the value in R0 to [R1], R1 should have the address of variable c. However, we should load the value of c to the register file before we store the value in R0 to [R1], otherwise we will lose the value of variable c. Thus, we should load the value of c to a new register, so we will need at least three registers.