Recitation#12: X86 assembly

CS232 Spring 2021

When: April 16 at 2:00 pm

---

1. Write the assembly language version of swapping two integers using the following assumptions. Hint: You need a maximum of 8 lines (you can do it in less, too).

   eax contains the first parameter (int *a)

   edx contains the second parameter (int *b)

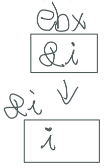
```
void swap (int *a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

```
movl (%eax), %ecx  =*a
movl (%edx), % ebx  =*b
movl  %ecx ,(% edx)
movl  % ebx, % eax
```

3
100
2
250
eax
100
ecx
2
edx
200
ebx
3

2. Assume the address of variable `i` is in register `%ebx`, given the following assembly code

```
movl (%ebx), %ecx
addl %ecx, %ecx
movl %ecx, (%ebx)
```



   Based on the 3-step common sequence of instructions explained in lecture, write some C code to match this assembly code

① move value of i into %ecx   int i;

② add what's in %ecx to %ecx & store there

i = i + i

③ Move what's inside %ecx back to &i.

3. Assume there are two integer variables **num1** and **num2** at addresses **0x8051004** and **0x8051000** respectively. The following is the assembly code for some arithmetic expression involving these two integer variables. Also assume that the final result of this arithmetic expression is stored in an integer variable named **result** at memory address **0x8050FF0.** The temporary variables temp1, temp2, and temp3 (that are used for computing the final result) are stored at locations 0x8050FFC, 0x8050FF8, and 0x8050FF4 respectively. Your task is to find out the following:
   a. arithmetic expressions for the variables temp1, temp2, temp3, and result
   b. final value of the variable result
   c. final values in registers %eax and %edx

   The value in the register ebx is given below: **%ebx = 0x8051004**

```
movl   $3, (%ebx)
```

Num1 = 3

```
movl  $7, -4(%ebx)       Num 2 = 7
movl  (%ebx), %eax       eax : 3
imull   -4(%ebx), %eax   eax : 3·7 = 21
movl  %eax, -8(%ebx)     temp1 = 21
movl  -4(%ebx), %eax     eax = 7
movl $0, %edx            edx = 0
idivl   (%ebx)           7/3 = →
movl  %eax, -12(%ebx)    temp 2 = 2
movl  (%ebx), %eax       eax : 3
movl $0, %edx            edx : 0      03/7
idivl   -4(%ebx)         3/7
movl  %edx, -16(%ebx)    temp 3 = 3
movl  -8(%ebx), %edx     edx : 21
movl  -12(%ebx), %eax    eax : 2
addl  %edx, %eax         21 + 2 = 23 : eax
subl  -16(%ebx), %eax    23 - 3 = 20 : eax
movl  %eax, -20(%ebx)    result = 20
```

eax ~~21~~ ~~7~~ 20

edx 0 ebx 3

eax Quotient 2 edx remainder 1

eax 0 edx ~~3~~ → 21

temp1 : num1 * num2

temp2 : num2 / num3

temp3 : num1 % num2

The C code snippet that produced the above assembly is given below. You should fill in the arithmetic expressions for temp1, temp2, temp3, and result.

int num1 = 3; 0x8051004 → %ebx

int num2 = 7; 0x8051000 → -4(%ebx)

int temp1 = 0x8050FFC; → -8(%ebx)

int temp2 = 0x8050FF8; -12(%ebx)

int temp3 = 0x8050FF4; -16(%ebx)

int result = 0x805FF0; -20(%ebx)

Final value of the variable **result** = 20

Final value in register **%eax** = 20

Final value in register **%edx** = 21