CSC231 - Assembly

Week #5

Dominique Thiébaut
dthiebaut@smith.edu
Review: Our Goal was...

```c
int x, y, sum;
x = 3;
y = 5;
sum = x + y;
```
Same with Short Ints

Translate this into Assembly

```c
short x, y, sum;
x = 3;
y = 5;
sum = x + y;
```
Same with Bytes

```c
byte x, y, sum;
x = 3;
y = 5;
sum = x + y;
```

Translate this into Assembly
int A[3];
A[0] = 3;
A[1] = 5;
extern  _printInt
extern   _println

section .data
a   dw      1
b   dw      2
c   dw      3
sum  dw      0
msg   db      "hello!",10

section .text
   global   _start
_start:
   mov     eax, dword[a]
   add     eax, dword[b]
   add     eax, dword[c]
   mov     dword[sum],eax
   call    _printInt
   call    _println

   mov     ecx, msg
   mov     edx, 7
   mov     eax, 4
   mov     ebx, 1
   int 0x80

What is the output of this program?
cs231a@marax:~ $ nasm -f elf week5exercise.asm
cs231a@marax:~ $ ld -melf_i386 week5exercise.o 231Lib.o -o week5exercise
cs231a@marax:~ $ ./week5exercise
327686
llo!

why?
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>db</td>
<td>99</td>
</tr>
<tr>
<td>b</td>
<td>db</td>
<td>88</td>
</tr>
<tr>
<td>c</td>
<td>dw</td>
<td>77</td>
</tr>
<tr>
<td>d</td>
<td>dw</td>
<td>66</td>
</tr>
<tr>
<td>e</td>
<td>dd</td>
<td>55</td>
</tr>
<tr>
<td>f</td>
<td>db</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td></td>
<td>33</td>
</tr>
<tr>
<td></td>
<td></td>
<td>22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1f</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00</td>
</tr>
</tbody>
</table>

section .data

section .text

Exercise

reconstruct the declarations of a, b, c, d, e and f.

hex

typical midterm question!
Follow a step by step execution of the program
section .data

a  dd 3
b  dd 5
sum dd 0

;;; code area

section .text

_start:  mov eax, dword[a]  ;eax <-- a
add eax, dword[b]  ;eax <-- eax+b = a+b
sub eax, 1     ;eax <-- eax-1 = a+b-1
mov dword[sum], eax ;sum <-- eax = a+b-1

;;; exit()

mov eax,1
mov ebx,0
int 0x80 ; final system call
\[
\begin{array}{c|c}
\text{a} & \text{dd} & 3 \\
\text{b} & \text{dd} & 5 \\
\text{sum} & \text{dd} & 0 \\
\end{array}
\]

100 \text{ mov eax, dword[a]} \\
105 \text{ add eax, dword[b]} \\
10A \text{ sub eax, 1} \\
10E \text{ mov dword[sum], eax}

\[
\begin{align*}
\text{eax} \\
\text{ebx} \\
\text{ecx} \\
\text{edx} \\
\text{eip}
\end{align*}
\]
Tick!

```
a dd 3
b dd 5
sum dd 0
```

```
mov eax, dword[a]
add eax, dword[b]
sub eax, 1
mov dword[sum], eax
```
D. Thiebaut, Computer Science, Smith College
Frequency: 3.2 GHz

$\frac{1}{3.2 \text{ GHz}} = 0.3125 \text{ ns}$
Arduino

Clock speed: **16 MHz**

~1/200 speed of Pentium

;hello.asm
;
; turns on an LED which is connected to PB5 (digital out 13)

.include "./m328Pdef.inc"

    ldi r16,0b00100000
    out DDRB,r16
    out PortB,r16
Start:
    rjmp Start

Clock speed: 1.4 GHz
~1/3 speed of Pentium

```c
/* -- first.s */
/* This is a comment */
.global main /* 'main' is our entry point and must be global */
main:       /* This is main */
    mov r0, #2 /* Put a 2 inside the register r0 */
    bx lr    /* Return from main */
```

Decimal
Binary
Base 3
Hexadecimal
Conversion
# prompts user for an integer
# decomposes the integer into binary

x = int( input( "> " ) )
binary = ""

while True:
    if x==0:
        break
    remainder = x % 2
    quotient  = x // 2

    if remainder == 0:
        binary = "0" + binary
    else:
        binary = "1" + binary

    print( "%5d = %5d * 2 + %d    quotient=%5d remainder=%d binary=%16s" % (x, quotient, remainder, quotient, remainder, binary) )

    x = quotient
# prompts user for an integer
# decomposes the integer into binary

x = int( input( "> " ) )
binary = ""

while True:
    if x==0:
        break
    remainder = x % 2
    quotient  = x // 2
    if remainder == 0:
        binary = "0" + binary
    else:
        binary = "1" + binary

print( "%5d = %5d * 2 + %d quotient=%5d remainder=%d binary=%16s"
      % (x, quotient, remainder, quotient, remainder, binary ) )

x = quotient
Exercises