Logistics

• Lab report due Wednesday 11:55 p.m.

• Windows users, install Putty on your computer (putty.org). Mac users, all set.

• Wednesday, emacs/nasm lab

• This Thursday, 5:15 p.m., Special lab on emacs and login in to Linux class accounts with David Marshall, FH241
Today...

- Review of demolition lab
- The main players (CPU, RAM, hard drive)
- Game
- Simplified Simulator
Today...

- Review of demolition lab
- The main players (CPU, RAM, hard drive)
- Game
- Simplified Simulator
Last Week…

Review
Main Players

Pentium
Main Players

Pentium

RAM
Main Players

Pentium

RAM

Disk Drive
Main Players

Pentium

RAM

Disk Drive

hello.asm
Main Players

- Pentium
- RAM
- Disk Drive

hello.asm
display "Hello World!" on the screen

To assemble, link, and run:
    nasm -f elf hello.asm
    ld -m elf_i386 -o hello hello.o
    ./hello

section .data
Hello           db      "Hello World!", 10, 10
HelloLen        equ     $-Hello

section .text
global  _start

_start:

;;; print message
    mov     eax, 4          ; write
    mov     ebx, 1          ; stdout
    mov     ecx, Hello      ; address of message to print
    mov     edx, HelloLen   ; # of chars to print
    int     0x80

;;; exit
    mov     ebx, 0
    mov     eax, 1
    int     0x80
Main Players

Pentium

RAM

Disk Drive

hello.asm

nasm

hello.o
Main Players

- Pentium
- RAM
- Disk Drive

Compilers:
- nasm
- ld
Main Players

Pentium

RAM

Disk Drive

hello.asm

ld

.nasm

hello.o

hello

./hello
Main Players

Pentium

RAM

Disk Drive

hello.asm

nasm

hello.o

ld

./hello

hello
Main Players

- Pentium
- RAM
- Disk Drive

hello.asm

nasm

ld

./hello
Main Players

- Pentium
- RAM
- Disk Drive

Compilation Process:

1. `nasm` hello.asm -> hello.o
2. `ld` hello.o -> hello
Linux Accounts

- cs231a-xx

- **Keep** top part, **sign** the bottom part and **return it**

- We will use them on **Wednesday** (in-class lab)

- Windows Users, please install **Putty** on your computer:
  
  - [https://www.putty.org](https://www.putty.org)
A Microprocessor Simulator
The Processor

Atmega16U2 (USB interface)

USB interface

Voltage regulator

Power in

Power LED

Atmega328P

https://i.stack.imgur.com/uxIPv.jpg


http://www.scind.org/content_images/full/1445359752New_Processor.jpg
Let's Play a Game
Rules #1

00 Really?
01 Hello
02 How are you?
03 Did you enjoy breakfast?
04 Was it good?
05 Have you stared on the homework?
06 Do you like apples?
07 Do you like bananas?
08 Goodbye!

00 No
01 Yes
02 Good
03 Bad
04 A tiny bit
05 A lot
06 Sometimes
07 Hello!
08 Sad
09 Goodbye!
Really?
Hello
How are you?
Did you enjoy ...
Was it good?
Have you stared on the homework?
Do you like ...
Goodbye!
Apples
Bananas
Breakfast

No
Yes
Good
Bad
A tiny bit
A lot
Sometimes
Hello!
Sad
Goodbye!
How could we setup a conversation with the whole class?
00 Really?
01 Hello
02 How are you?
10 Did you enjoy …?
04 Was it good?
05 Have you stared on the homework?
11 Do you like …
08 Goodbye!
12 Apples
13 Bananas
14 Breakfast

00 No
01 Yes
02 Good
03 Bad
04 A tiny bit
05 A lot
06 Sometimes
07 Hello!
08 Sad
09 Goodbye!
Exercise
Recreate the original conversations
On Infinite Verbal Loops...
Important Concepts

• Codes, codes, codes

• Numbers can represent questions, as well as answers

• The same number can have different meanings, depending on context

• With the right choice of questions and answers, coded as numbers, one could have a rich conversation!
Important Concepts Relating to Computers

• Only numbers can be stored in memory

• The memory is made of bits

• Bits represent binary digits

• The memory contains only binary numbers

• Using codes, the numbers can be used to represent characters (a, b, A, B, 0, 1, #, &…)

• Using codes, the numbers can be used to represent actions to be executed by the processor

• Using codes, the binary numbers can be used to represent decimal numbers, integer, or real (1, -2, 10, 3.14159…)
# The ASCII Table

**American Standard Code for Information Interchange**

<table>
<thead>
<tr>
<th>ASCII value</th>
<th>Character</th>
<th>Control character</th>
<th>ASCII value</th>
<th>Character</th>
<th>ASCII value</th>
<th>Character</th>
<th>ASCII value</th>
<th>Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>(null)</td>
<td>NUL</td>
<td>032</td>
<td>(space)</td>
<td>064</td>
<td>@</td>
<td>096</td>
<td></td>
</tr>
<tr>
<td>001</td>
<td>⌐</td>
<td>SOH</td>
<td>033</td>
<td>!</td>
<td>065</td>
<td>A</td>
<td>097</td>
<td>a</td>
</tr>
<tr>
<td>002</td>
<td>⤀</td>
<td>STX</td>
<td>034</td>
<td>&quot;</td>
<td>066</td>
<td>B</td>
<td>098</td>
<td>b</td>
</tr>
<tr>
<td>003</td>
<td>⬫</td>
<td>ETX</td>
<td>035</td>
<td>#</td>
<td>067</td>
<td>C</td>
<td>099</td>
<td>c</td>
</tr>
<tr>
<td>004</td>
<td>⬬</td>
<td>EOT</td>
<td>036</td>
<td>$</td>
<td>068</td>
<td>D</td>
<td>100</td>
<td>d</td>
</tr>
<tr>
<td>005</td>
<td>⬬</td>
<td>ENQ</td>
<td>037</td>
<td>%</td>
<td>069</td>
<td>E</td>
<td>101</td>
<td>e</td>
</tr>
<tr>
<td>006</td>
<td>⬬</td>
<td>ACK</td>
<td>038</td>
<td>&amp;</td>
<td>070</td>
<td>F</td>
<td>102</td>
<td>f</td>
</tr>
<tr>
<td>007</td>
<td>⬬</td>
<td>BEL</td>
<td>039</td>
<td>'</td>
<td>071</td>
<td>G</td>
<td>103</td>
<td>g</td>
</tr>
<tr>
<td>008</td>
<td>⬬</td>
<td>BS</td>
<td>040</td>
<td>(</td>
<td>072</td>
<td>H</td>
<td>104</td>
<td>h</td>
</tr>
<tr>
<td>009</td>
<td>⬬</td>
<td>HT</td>
<td>041</td>
<td>)</td>
<td>073</td>
<td>I</td>
<td>105</td>
<td>i</td>
</tr>
<tr>
<td>010</td>
<td>⬬</td>
<td>LF</td>
<td>042</td>
<td>*</td>
<td>074</td>
<td>J</td>
<td>106</td>
<td>j</td>
</tr>
<tr>
<td>011</td>
<td>⬬</td>
<td>VT</td>
<td>043</td>
<td>+</td>
<td>075</td>
<td>K</td>
<td>107</td>
<td>k</td>
</tr>
<tr>
<td>012</td>
<td>⬬</td>
<td>FF</td>
<td>044</td>
<td>-</td>
<td>076</td>
<td>L</td>
<td>108</td>
<td>l</td>
</tr>
<tr>
<td>013</td>
<td>⬬</td>
<td>CR</td>
<td>045</td>
<td>.</td>
<td>077</td>
<td>M</td>
<td>109</td>
<td>m</td>
</tr>
<tr>
<td>014</td>
<td>⬬</td>
<td>SO</td>
<td>046</td>
<td>/</td>
<td>078</td>
<td>N</td>
<td>110</td>
<td>n</td>
</tr>
<tr>
<td>015</td>
<td>⬬</td>
<td>SI</td>
<td>047</td>
<td>0</td>
<td>079</td>
<td>O</td>
<td>111</td>
<td>o</td>
</tr>
<tr>
<td>016</td>
<td>⬬</td>
<td>DLE</td>
<td>048</td>
<td>D</td>
<td>080</td>
<td>P</td>
<td>112</td>
<td>p</td>
</tr>
<tr>
<td>017</td>
<td>⬬</td>
<td>DC1</td>
<td>049</td>
<td>1</td>
<td>081</td>
<td>Q</td>
<td>113</td>
<td>q</td>
</tr>
<tr>
<td>018</td>
<td>⬬</td>
<td>DC2</td>
<td>050</td>
<td>2</td>
<td>082</td>
<td>R</td>
<td>114</td>
<td>r</td>
</tr>
<tr>
<td>019</td>
<td>⬬</td>
<td>DC3</td>
<td>051</td>
<td>3</td>
<td>083</td>
<td>S</td>
<td>115</td>
<td>s</td>
</tr>
<tr>
<td>020</td>
<td>⬬</td>
<td>DC4</td>
<td>052</td>
<td>4</td>
<td>084</td>
<td>T</td>
<td>116</td>
<td>t</td>
</tr>
<tr>
<td>021</td>
<td>⬬</td>
<td>NAK</td>
<td>053</td>
<td>5</td>
<td>085</td>
<td>U</td>
<td>117</td>
<td>u</td>
</tr>
<tr>
<td>022</td>
<td>⬬</td>
<td>SYN</td>
<td>054</td>
<td>6</td>
<td>086</td>
<td>V</td>
<td>118</td>
<td>v</td>
</tr>
<tr>
<td>023</td>
<td>⬬</td>
<td>ETB</td>
<td>055</td>
<td>7</td>
<td>087</td>
<td>W</td>
<td>119</td>
<td>w</td>
</tr>
<tr>
<td>024</td>
<td>⬬</td>
<td>CAN</td>
<td>056</td>
<td>8</td>
<td>088</td>
<td>X</td>
<td>120</td>
<td>x</td>
</tr>
<tr>
<td>025</td>
<td>⬬</td>
<td>EM</td>
<td>057</td>
<td>9</td>
<td>089</td>
<td>Y</td>
<td>121</td>
<td>y</td>
</tr>
<tr>
<td>026</td>
<td>⬬</td>
<td>SUB</td>
<td>058</td>
<td>;</td>
<td>090</td>
<td>Z</td>
<td>122</td>
<td>z</td>
</tr>
<tr>
<td>027</td>
<td>⬬</td>
<td>ESC</td>
<td>059</td>
<td>...</td>
<td>091</td>
<td>[</td>
<td>123</td>
<td>{</td>
</tr>
<tr>
<td>028</td>
<td>⬬</td>
<td>FS</td>
<td>060</td>
<td>&lt;</td>
<td>092</td>
<td>\</td>
<td>124</td>
<td>\</td>
</tr>
<tr>
<td>029</td>
<td>⬬</td>
<td>GS</td>
<td>061</td>
<td>=</td>
<td>093</td>
<td>]</td>
<td>125</td>
<td>}</td>
</tr>
<tr>
<td>030</td>
<td>⬬</td>
<td>RS</td>
<td>062</td>
<td>&gt;</td>
<td>094</td>
<td>^</td>
<td>126</td>
<td>~</td>
</tr>
<tr>
<td>031</td>
<td>⬬</td>
<td>US</td>
<td>063</td>
<td>?</td>
<td>095</td>
<td>–</td>
<td>127</td>
<td>☐</td>
</tr>
</tbody>
</table>

Hand Calculator Example
Assembly Language
With a Simulator
The Key Players

Processor

AC
PC

Memory
Types of Operation Supported

- LOAD
- STORE
- ADD
- HALT
LOAD instruction (memory)
LOAD instruction (immediate)

Example
STORE instruction (memory)
Program #1

Processor

AC

PC

LOAD 1

STORE [9]

HALT

11

10

9

8

7

6

5

4

3

2

1

0

103
Program #2

Processor

AC

PC

+  

11

10

9

8

7

6

5

4

3

2

1

0

HALT
STORE [9]
ADD 1
LOAD [9]
Example

ADD instruction (memory)
ADD instruction (immediate)
Write an assembly language program that takes the numbers stored at Addresses 9, and 10 adds them together, and stores the sum back at Address 11.
Example

JUMP instruction

Processor

AC
PC

+  

11
10
9  
8  
7  
6  
5  
4  
3  
2  
1  
0

Jump 0
Store [9]
Add 1
Load [9]
HALT instruction

Example

Processor

AC

PC

+
Computer Simulator


http://www.science.smith.edu/dftwiki/media/simulator/
Resources

- Link to the simulator
- Link to the documentation
Exercise

Write an assembly language program that uses 3 variables in memory, containing 30, 40, and 0, respectively. The program will compute the sum of the first 2 and store the result in the 3th variable. Run your program on the simulator.