How are Integers Stored in Memory?
msg1 db "Hello"
Have we seen this before?
Endianness

• **Little-Endian** Processors
  • Intel Pentium
  • Apple's original MOS 6502
  • Zilog Z80

• **Big-Endian** Processors
  • Motorola 68000
  • Atmel AVR32 (Arduino)
Current generation ARM processors (from ARM6 onwards) have the option of operating in either little-endian or big-endian mode. These terms refer to the way in which multi-byte quantities, such as 32-bit words, are stored in a byte-addressed memory.

http://netwinder.osuosl.org/pub/netwinder/docs/arm/Apps04vC.html
Implications

- Serialization of data

http://ubjson.org/#endian
Pentium Data Registers
Pentium Registers

eax
ebx
ecx
edx
Pentium Registers

<table>
<thead>
<tr>
<th>eax</th>
<th>ax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebx</td>
<td>bx</td>
</tr>
<tr>
<td>ecx</td>
<td>cx</td>
</tr>
<tr>
<td>edx</td>
<td>dx</td>
</tr>
</tbody>
</table>
Pentium Registers

<table>
<thead>
<tr>
<th>eax</th>
<th>ah</th>
<th>al</th>
<th>ax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebx</td>
<td>bh</td>
<td>bl</td>
<td>bx</td>
</tr>
<tr>
<td>ecx</td>
<td>ch</td>
<td>cl</td>
<td>cx</td>
</tr>
<tr>
<td>edx</td>
<td>dh</td>
<td>dl</td>
<td>dx</td>
</tr>
</tbody>
</table>
Think of **ah** and **al**
as boxes inside
a bigger one
called **ax**, and **ax** as
half of a bigger
box still, called **eax**.
Declaring Variables
• **db**: define byte storage

• **dw**: define word storage

• **dd**: define double-word storage
### Examples: db

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>msg</td>
<td>db</td>
<td>&quot;Hello&quot;, 10</td>
</tr>
<tr>
<td>a</td>
<td>db</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>db</td>
<td>'H' ; also 72 or 0x48</td>
</tr>
<tr>
<td>c</td>
<td>db</td>
<td>255</td>
</tr>
<tr>
<td>d</td>
<td>db</td>
<td>0x80</td>
</tr>
<tr>
<td>Variable</td>
<td>dw</td>
<td>Value</td>
</tr>
<tr>
<td>----------</td>
<td>----</td>
<td>---------</td>
</tr>
<tr>
<td>x</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>z</td>
<td></td>
<td>255</td>
</tr>
<tr>
<td>t</td>
<td></td>
<td>0x1234</td>
</tr>
</tbody>
</table>
### Examples: `dd`

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha</td>
<td><code>dd 0</code></td>
</tr>
<tr>
<td>beta</td>
<td><code>dd 255</code></td>
</tr>
<tr>
<td>gamma</td>
<td><code>dd 0x12345678</code></td>
</tr>
</tbody>
</table>
We stopped here last time...
Summary of important concepts just seen

- Numbers
- Op Codes
- Machine Language
- Hexadecimal
- Executable Files
Return to the mov instruction

mov dest, source
section .data
lf       db      10
ch       db      0
a        dw      0x1234
b        dw      0
x        dd      0
y        dd      0x12345678

section .text
; put lf in al

eax          ah al    ax
ebx          bh bl    bx
ecx          ch cl    cx
edx          dh dl    dx
```assembly
section .data
lf       db      10
ch       db      0
a        dw      0x1234
b        dw      0
x        dd      0
y        dd      0x12345678

section .text
; put al in ch
```
section .data
lf db 10
ch db 0
a dw 0x1234
b dw 0
x dd 0
y dd 0x12345678

section .text
; put a in bx

; put bx in b

; put bx in ax

; put 0 in cx
section .data
lf db 10
ch db 0
a dw 0x1234
b dw 0
x dd 0
y dd 0x12345678

section .text
; put x in eax
; put y in ecx
; put ecx in edx
; put ex into y

Test Cases
section .data
lf       db      10
ch       db      0
a        dw      0x1234
b        dw      0
x        dd      0
y        dd      0x12345678

section .text
; put 0 in ah
; put 3 in cx
; put 5 in edx
; put 0x12345678 into eax

<table>
<thead>
<tr>
<th>eax</th>
<th>ah</th>
<th>al</th>
<th>ax</th>
</tr>
</thead>
<tbody>
<tr>
<td>ebx</td>
<td>bh</td>
<td>bl</td>
<td>bx</td>
</tr>
<tr>
<td>ecx</td>
<td>ch</td>
<td>cl</td>
<td>cx</td>
</tr>
<tr>
<td>edx</td>
<td>dh</td>
<td>dl</td>
<td>dx</td>
</tr>
</tbody>
</table>
We understand mov!
The add instruction Revisited

add dest, source
The add instruction Revisited

```
add dest, source

reg8  reg16  reg32
mem8  mem16  mem32
imm8  imm16  imm32
```
The add instruction Revisited

add dest, source

reg8
reg16
reg32
mem8
mem16
mem32

reg8
reg16
reg32
mem8
mem16
mem32
imm8
imm16
imm32
The add instruction Revisited

Sizes have to match
section .data
lf        db      10
ch        db      0
a         dw      0x1234
b         dw      0
x         dd      0
y         dd      0x12345678

section .text
; add 3 to ch

; add 100 to b

; add -1 to edx

; add x to y
Reminder: Our Goal was...

We translated this into Assembly

```c
int x, y, sum;
x = 3;
y = 5;
sum = x + y;
```
section .data
a  db      10
b  db      0
c  dw      0x1234
d  dw      0
e  dd      0
f  dd      0x12345678

section .text

Swap a and b. Then c and d. Then e and f.
section .data
a db 10
b db 0
c dw 0x1234
d dw 0
e dd 0xcdef
f dd 0x12345678

section .text

Set the least significant byte of e and f to 00.
<table>
<thead>
<tr>
<th>Section</th>
<th>Declaration</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>.data</td>
<td>a</td>
<td>db</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>db</td>
</tr>
<tr>
<td></td>
<td>c</td>
<td>dw</td>
</tr>
<tr>
<td></td>
<td>d</td>
<td>dw</td>
</tr>
<tr>
<td></td>
<td>e</td>
<td>dd</td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>db</td>
</tr>
<tr>
<td>.text</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Exercise

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

Typical midterm question!
```
<table>
<thead>
<tr>
<th>section .data</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>db</td>
<td></td>
<td></td>
<td></td>
<td>99</td>
</tr>
<tr>
<td>b</td>
<td>db</td>
<td></td>
<td></td>
<td></td>
<td>88</td>
</tr>
<tr>
<td>c</td>
<td>dw</td>
<td></td>
<td></td>
<td></td>
<td>77</td>
</tr>
<tr>
<td>d</td>
<td>dw</td>
<td></td>
<td></td>
<td></td>
<td>66</td>
</tr>
<tr>
<td>e</td>
<td>dd</td>
<td></td>
<td></td>
<td></td>
<td>55</td>
</tr>
<tr>
<td>f</td>
<td>db</td>
<td></td>
<td></td>
<td></td>
<td>44</td>
</tr>
</tbody>
</table>

section .text
```

Exercise

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

D. Thiebaut, Computer Science, Smith College

**typical midterm question!**
Following the step by step execution of the program