Assembly Directives

Ned Nedialkov

McMaster University
Canada

SE 3F03
January 2014

Outline

Directives

- A directive is an instruction to the assembler, not the CPU
- A directive is not an executable instruction
- A directive can be used to
  - define a constant
  - define memory for data
  - include source code & other file
- They are similar to C's `include` and `define`

equ directive

- `symbol` **equ** `value`
- Defines a symbol
- Cannot be redefined later

% directive

- `%define symbol value`
- Similar to `#define` in C
- E.g.

```
%define N 100
;
mov eax, N
```

- Can be redefined

Data formats

name	abbreviation	size
byte	b	
word	w	2 bytes
double word	d	4 bytes
quad word	q	8 bytes
10 bytes	t	10 bytes

Data directives

- Define room for uninitialized or uninitialized data
- E.g

```
L1   db 0        ;defines a byte and initializes to 0
L2   dw FF0Fh    ;define a word and initialize to FF0Fh
L3   db "A"      ;byte initialized with the ASCII value of A
L4   dw 100      ;reserve space for 100 double words
L5   times 100 db 0 ;reserve 100 bytes and initialize to 0
L6   db "s","t","r","i","n","g",0 ;defines "string"
L7   db 'string',0                ;same as above
```

- Double and single quotes are treated the same

- `RESx` directive; `x` is one of `b,w,d,q,t`
- `Dx` directive; `x` is one of `b,w,d,q,t`

Examples

```
mov      al, [L2] ;move a byte at L2 to al
mov      eax, L2  ;move the address of L2 into eax
mov      [L1], ah ;move ah into the first byte pointed to by L1
mov      eax, 5
add      [L2], eax ;double word at L2
                   ;containing [L2]+eax
mov      [L2], 1   ;does not work, why?
mov dword [L2], 1 ;works, why?
```

Big endian and little endian representations

- Big endian: the most significant byte is stored first
  - E.g. `12345678` is stored as `12|34|56|78`
- Little endian: the least significant byte is stored first
  - E.g. `12345678` is stored as `78|56|34|12`
- x86 uses little endian

```
#include<stdio.h>
int main()
{
  unsigned short word = 0x1234;
  unsigned char *p = (unsigned char *) &word;
  if ( p[0] == 0x12 )
    printf ("Big␣Endian␣Machine\n");
  else
    printf("Little␣Endian␣Machine\n");
  return 0;
}
```