```
; Illustrate various addressing
; modes and the code that they generate with mov
; The general rules and their exceptions follow:
; The opcode determines direction of move,
; for example 89 is from the register specified in
; the reg field of mod-reg-r/m, 8B to.
; register encoding is as follows:
; 000 eax, 001 ecx, 010 edx, 011 ebx,
; 100 esp, 101 ebp, 110 esi, 111 edi
; the mod-reg-r/m (mode register register/memory) is one byte
; with mod 2-bits, reg, r/m each 3.
; r/m represents register operand, unless
;       exception(1): r/m==100 means SIB (scale-index-base) byte present
; mod: 00 - no displacement, address specified in r/m register
;       exception(2): r/m==101 means no registers,
;           address is 4-byte displacement
; mod: 01 - one byte displacement, address is r/m register contents
;           plus displacement
; mod: 10 - like 01, except 4-byte displacement
; mod: 11 - contents of reg r/m, or immediate value depending on opcode
; if the scale-index-base byte is present, that is when
;       mod == 00, 10, or 01 and r/m == 100
;       then the SIB byte is present and specifies addressing:
;       S is 2-bits, I and B are 3 bits.
;       address is
;           (contents of register I) left-shifted S bit positions
;            + contents of register B + displacement
;       exception(3): I==100 means no index is used.
;       exception(4): mod==00, B==101 means no base, 4-byte displacement
;       (last paragraph of nasm manual sec B.2.5 has misprint, should be
;           and base is 5)
;
;   Copy this from my public directory:
;   cp /home/mccaslinsj/pub/cmsc301/address.asm .
;   and assemble it with the list option:
;   make32 -l address.txt address.asm
;   to see what object code each of the following instruction generates.
;   (do not try to run this program, it doesn't do anything.)
section .data
x   times 200 dd 0 ; array of 200 dwords
a   dd  100
section .code
global main
main:
;   copy from a register
;   next 6 follow the general rules:
    mov ecx,edx             ; register
    mov [ecx],edx           ; register indirect
    mov [ebp],edx           ; base-pointer indirect
    mov [ebp+8],edx         ; byte-displacement from base, no index
    mov [a+ebp],edx         ; dword-displacement from base, no index
    mov [ecx+ebx],edx       ; index + register
;   exception(2), no registers involved in address
    mov [a],edx             ; displacement
;   next 5 are exception(1), SIB present
    mov [x+4*ecx],edx       ; scale*index plus displacement
    mov [a+ecx+ebp],edx     ; base index displacement
    mov [a+8*ecx+ebp],edx   ; base scale*index displacement
;   + exception(3)
    mov [esp+8],edx         ; byte-displacement, SIB, no index
;   + exception(4)
    mov [a+4*ebp],edx       ; dword-displacement, SIB, index, no base

;   mov [x+4*esp],edx       ; illegal, esp cannot be index

;   copy to a register
    mov edx,a               ; immediate (different op code)
;   next 6 follow the general rules:
    mov edx,ecx             ; register
    mov edx,[ecx]           ; register indirect
    mov edx,[ebp]           ; base-pointer indirect
    mov edx,[ebp+8]         ; byte-displacement from base, no index
    mov edx,[a+ebp]         ; dword-displacement from base, no index
    mov edx,[ecx+ebx]       ; index + register
;   exception(2), no registers involved in address
    mov edx,[a]             ; displacement
;   next 5 are exception(1), SIB present
    mov edx,[x+4*ecx]       ; scale*index plus displacement
    mov edx,[a+ecx+ebp]     ; base index displacement
    mov edx,[a+8*ecx+ebp]   ; base scale*index displacement
```

```
;   + exception(3)
    mov edx,[esp+8]         ; byte-displacement, SIB, no index
;   + exception(4)
    mov edx,[a+4*ebp]       ; dword-displacement, SIB, index, no base

;   mov edx,[x+4*esp]       ; illegal, esp cannot be index
```

Last Modified 11/30/2004 13:48:06