Programování na strojové úrovni

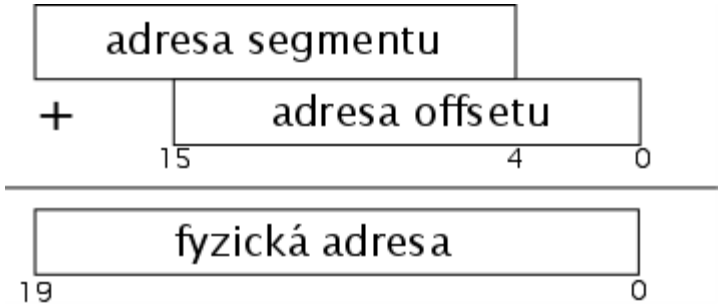
Cvičení 3: Registry a přístup do paměti

Tomáš Goldmann igoldmann@fit.vutbr.cz
www.fit.vutbr.cz/~igoldmann/courses

Rev.: 1A/2017

Paměťové režimy-reálný režim

- Pracuje s 20 bitovou fyzickou adresou.
- Při vytváření fyzické adresy se sečte hodnota adresy offsetu a segmentu, přičemž se adresa segmentu posune o 4 bity.
- V dnešní době při spuštění OS dojde k přepnutí do chráněného režimu.



Paměťové režimy - chráněný režim

Chráněný režim

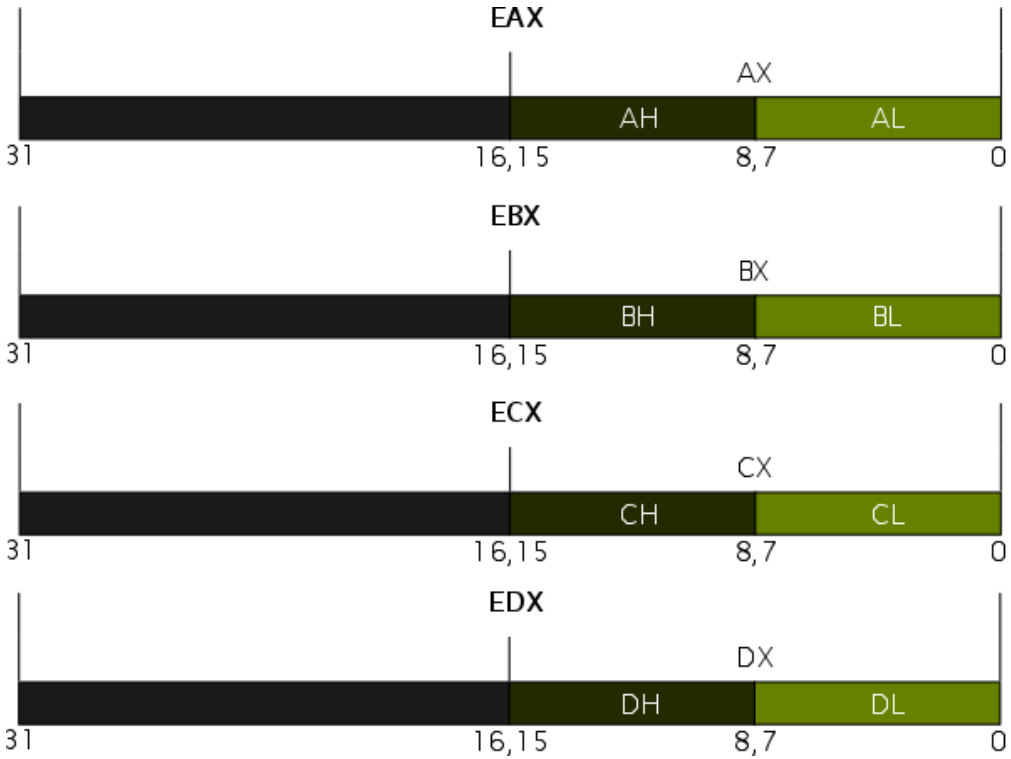
- V chráněném režimu se bázová adresa segmentu získává prostřednictvím indexování tabulky popisovače. Jako index se použije segment selector (nahrazuje segmentovou adresu).
- Výsledná adresa je daná součtem bázové adresy (32 bitů) a offsetové adresy (32 bitů).
- V dnešní době se segmentace využívá pouze omezeně.

Chráněný režim se stránkováním

Registry architektury x86

Obecné registry

- Registr **EAX** – akumulátor
- Registr **EBX** – bázový registr
- Registr **ECX** – kontrolní registr
- Registr **EDX** – datový registr

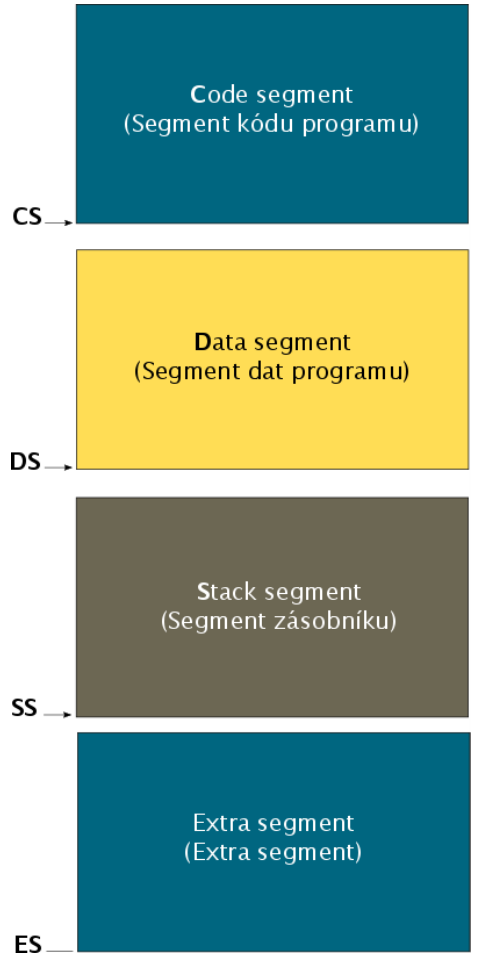


Registry architektury x86

Segmentové registry

Registry ukazující na oblasti, kde se nacházejí různé typy dat. Především pak kód programu, data programu a oblast zásobníku (CS,DS,SS,ES).

Tyto registry nejde přímo měnit.



Indexové registry

(E)SP – ukazatel na vrchol zásobníku

(E)BP – bázový ukazatel při práci se zásobníkem

(E)SI – registr pro index ukazující na zdroj

(E)DI – registr pro index ukazující na cíle

Příznakový registr EFLAGS

Overflow flag (OF) – výsledek aritmetické operace s celými čísly se znaménkem překročil povolený rozsah.

Carry flag (CF) – došlo k přenosu nebo výpůjčce na MSB.

Zero flag (ZF) – výsledek aritmetické operace je nulové hodnoty.

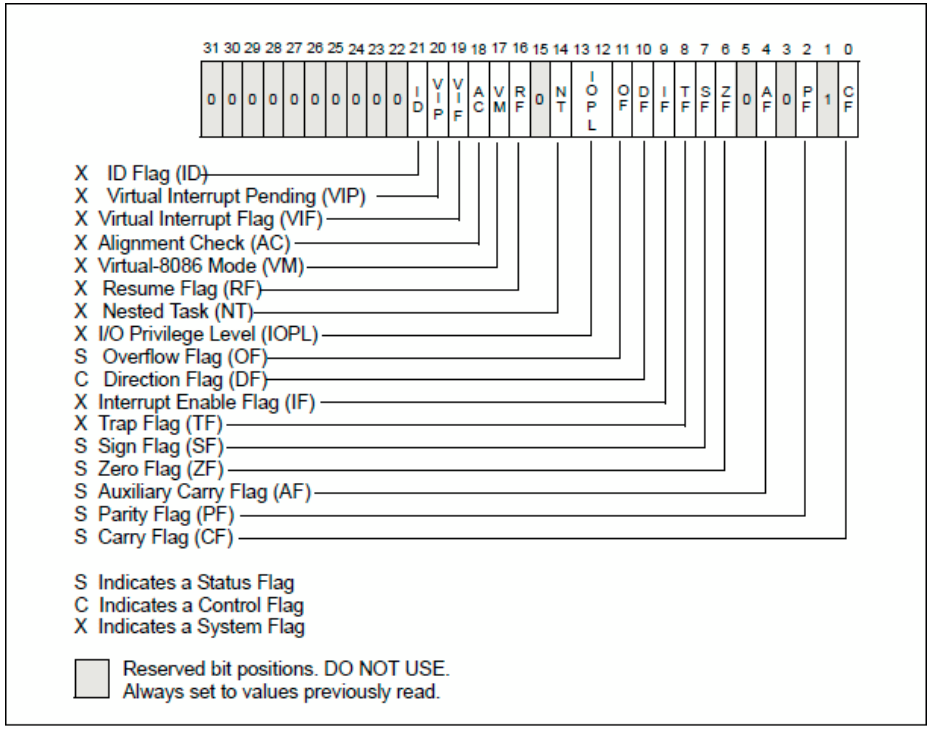
Auxilary carry flag (AF) – je nastaven, pokud dojde k přenosu z 3 bitu na 4 bit pro BCD kód.

Parity flag (PF) – je nastaven, jestliže počet jedniček výsledku poslední operace je sudý.

Sign flag (SF) – je nastaven, pokud je výsledek operace záporné číslo. (dvojkový doplněk)

Direction flag (DF) – příznak směru při zpracování řetězců

Trap flag (TF) – je nastavován při debugování

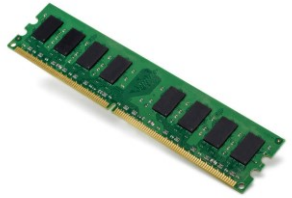


EFLAGS Register

Zdroj: http://www.c-jump.com/CIS77/ASM/Instructions/I77_0050_eflags.htm

Deklarace inicializovaných dat

Operační paměť



```
db    0x55                ; just the byte 0x55
db    0x55,0x56,0x57      ; three bytes in succession
db    'a',0x55            ; character constants are OK
db    'hello',13,10,'$'   ; so are string constants
dw    0x1234              ; 0x34 0x12
dw    'a'                 ; 0x61 0x00 (it's just a number)
dw    'ab'                ; 0x61 0x62 (character constant)
dw    'abc'               ; 0x61 0x62 0x63 0x00 (string)
dd    0x12345678          ; 0x78 0x56 0x34 0x12
dd    1.234567e20         ; floating-point constant
dq    0x123456789abcdef0  ; eight byte constant
dq    1.234567e20         ; double-precision float
dt    1.234567e20         ; extended-precision float
```

Source: http://www.nasm.us/doc/

Práce s pamětí

Adresování paměti

```
arrav   dd 89. 10. 67, 1, 4, 27, 12, 34, 86, 3
wordvar dd 123
```

Přesuneme hodnotu z paměti do EAX. Proč do EAX a ne do AX?

```
mov     eax, [wordvar]
```

Ukázka indexování druhého prvku. Proč 1*4?

```
mov     ebx, [array+1*4]
```

Práce s pamětí

Adresování paměti

Efektivní adresa operandu

EA=base + index*scaling_factor + displacement

Typy adresování

- Přímá adresa — mov al, [uk_byte]
- Nepřímá adresa — mov al, [uk_byte+ebx]
- Ukazatel přes bázový registr — mov ebx, uk_byte
mov al, [ebx]
- Ukazatel přes index registr — mov esi, uk_byte
mov al, [esi]

Příklad 1

- Výpočet obvodu obdélníku s 8 b a 16 b registrem

```
bits 32
section .data
        a db 100
        b dw 120
        O dw 0
section .text
start:
        xor __,__
        mov __, [a]
        add al, al
        mov __, [b]
        add bx,__
        add ax,bx
        mov [O],bx
ret
```

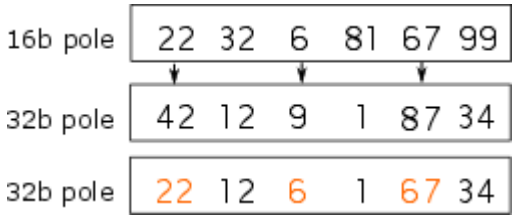
Paměť příklad 2

- Vypište hodnotu 1., 2., 4. prvku pole s tím, že offset po inicializaci budete měnit pouze aritmetickými operacemi ADD a SUB

```
bits 32
section .data
        pole db 10,20,30,40,50
section .text
start:
        mov bl,___
        mov al,______
        call WriteByte
        ___ ____,___
        ___ ____,___
        call WriteByte
        ___ ____,___
        ___ ____,___
        call WriteByte
ret
```

Úkoly

1. Inicializujte dvě pole o 6 hodnotách, kde první pole bude obsahovat 16 bitové hodnoty a druhé pole 32 bitové hodnoty. Nahraďte v prvním poli hodnoty se sudým indexem hodnotami z druhého pole.



2. Vygenerujte pomocí instrukcí ADD a SUB příznaky: ZF,SF,CF,PF,AF. Použijte 16 bitové registry.

3. Vytvořte si dvě pole o velikosti 10 prvků. Následně proveďte výpočet: pole1[i]= pole1[i]+ pole2[i]-4, i=0…9

4. Vytvořte pole a,b,c,d o libovolném počtu prvků. Následně vytvořte pole p_array s ukazateli na pole a,b,c,d. Vypište poslední prvek pole c pomocí p_array.