Programování na strojové úrovni

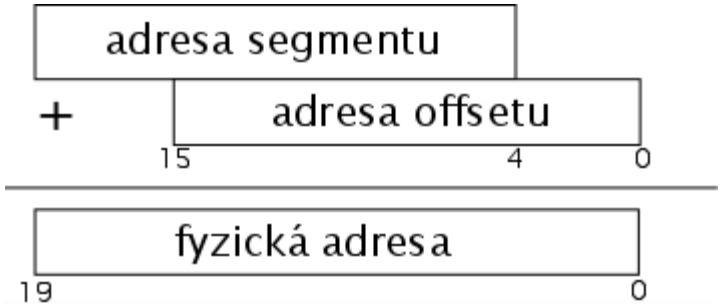
Cvičení 2: Registry a přístup do paměti

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

Rev.: 1A/2019

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

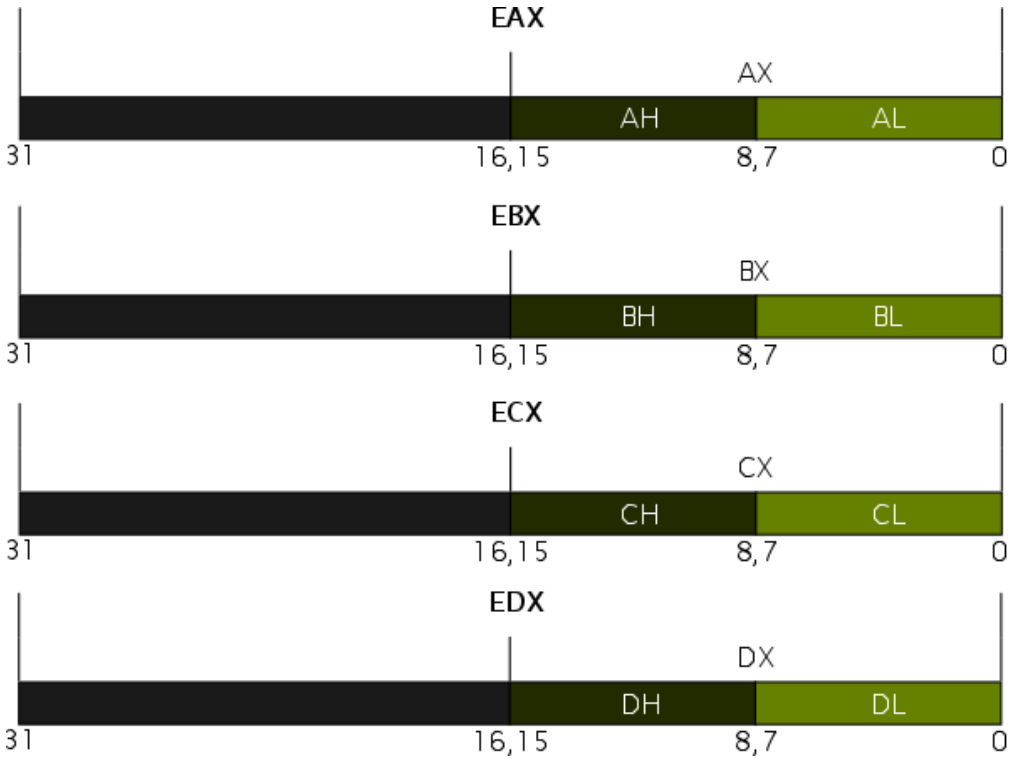
- V chráněném režimu se bázová adresa segmentu získává prostřednictvím indexování tabulky popisovače. Jako index se použije segment selector (nahrazuje segmentovou adresu).
- Výsledná adresa je daná součtem bázové adresy (32 bitů) a offsetové adresy (32 bitů).
- V dnešní době se segmentace využívá pouze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

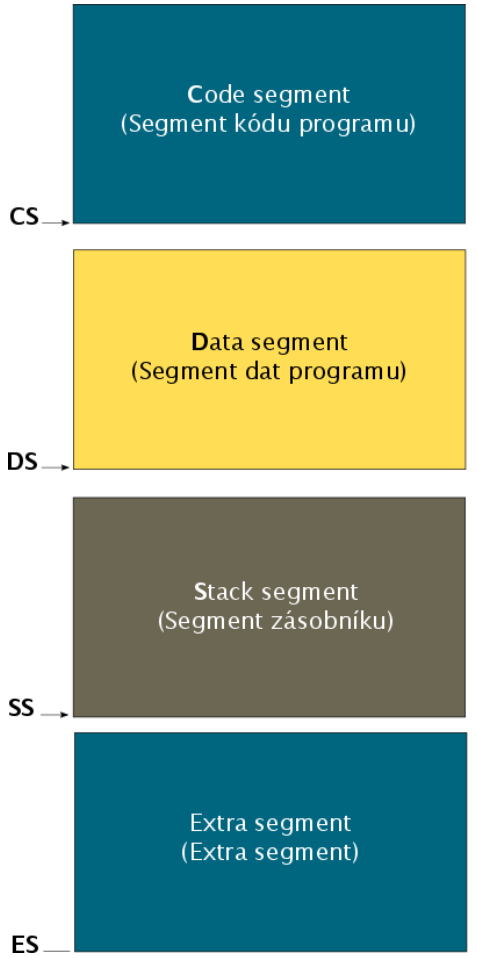


Registry architektury x86

- **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.



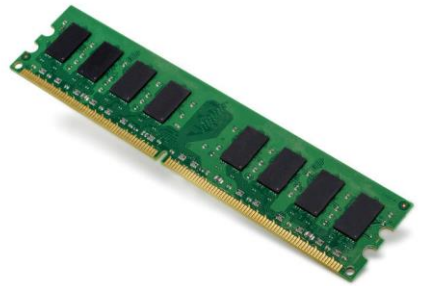
- **Indexové registry**

  (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

Deklarace inicializovaných dat

- **Inicializace hodnot pro .data v NASM**

```
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

array 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 prvku s indexem 1 (první prvek). 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]

Porovnání s jazykem C

```
section .data
pole db 0,0,0,0,0

section .text
start:
        mov eax, 1

        mov al, [pole+eax*1]
        mov ebx, eax
        add ebx, 3
        mov bl, [pole+ebx]

ret
```

```
char pole[5];

int main()
{
        int i=1;
        char al=pole[i];

        char bl=pole[i+3];
}
```

Napište si ☺ - Jak řešit příklady

Viz tabule

Napište si ☺ - Na co si dát pozor

Viz tabule

Příklad 1 - začínáme

- Sečťete obsah registru **al** a **bl** a výslednou hodnotu uložte do paměti se symbolickou adresou **isu.**

```
bits 32
section .data

___ __ _
section .text
start:
        mov al,120
        mov bl, 30

        ___ __, __
        mov __, __
ret
```

Příklad 2 - začínáme

- 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],ax
ret
```

Příklad 3 – indexování paměti

- Vypište hodnotu prvků pole s indexem 1.,2. a 4. 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 ecx,___
        mov al,______
        call WriteUInt8
        ___ ____,___
        ___ ____,___
        call WriteUInt8
        ___ ____,___
        ___ ____,___
        call WriteUInt8
ret
```

Příklady – samostatná práce

- Sečtěte 16 b a 32 b hodnotu z paměti a uložte ji do registru eax.

- Vygenerujte Segmentation fault