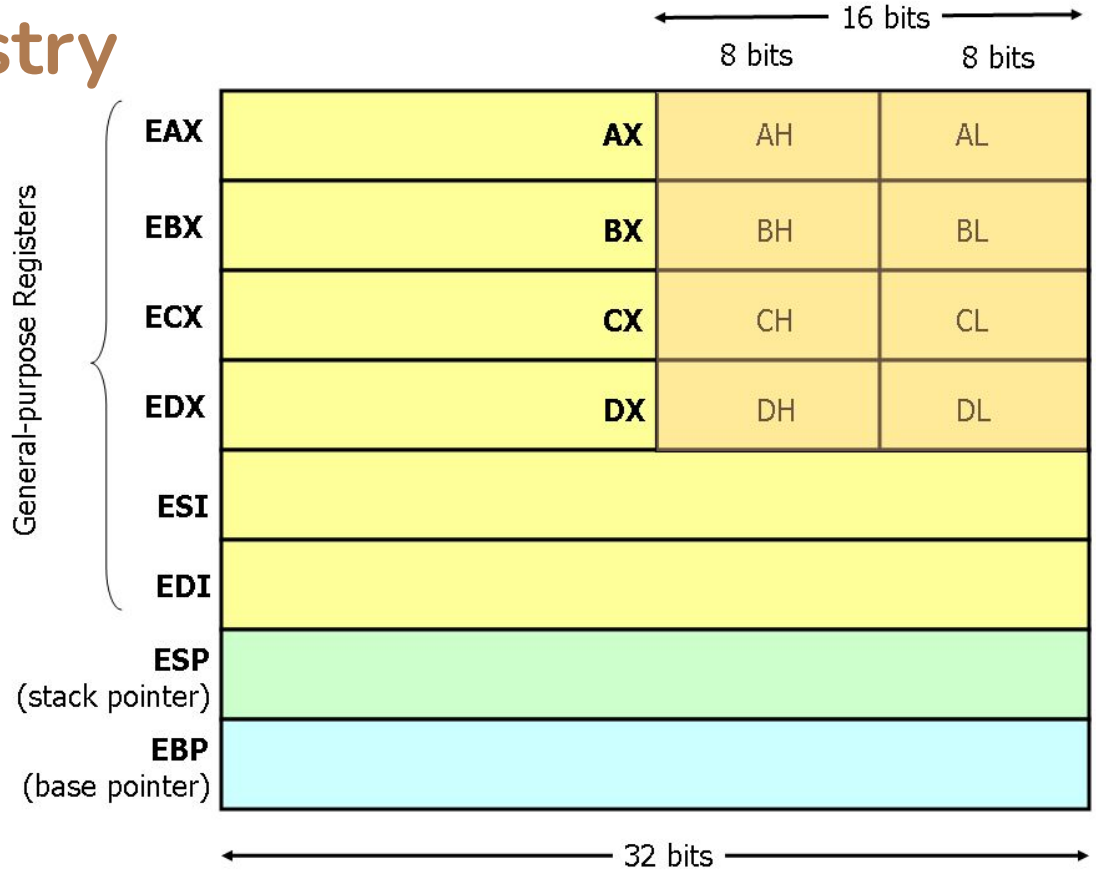
Paměť, pole

ISU cv 3

www.fit.vut.cz/person/isakin/public/isu

Registry


16 bits
8 bits
8 bits
EAX
AX
AH
AL
EBX
BX
BH
BL
ECX
CX
CH
CL
EDX
DX
DH
DL
ESI
EDI
ESP
(stack pointer)
EBP
(base pointer)
General-purpose Registers
32 bits

Globální proměnné

Datové segmety

- `section .data` – inicializovaný
  - proměnné se **známou** počáteční hodnotou
- `section .bss` – neinicializovaný
  - proměnné s **neznámou** počáteční hodnotou
  - Block Started by Symbol

- Lokální proměnné - ukládání do zásobníku

Globální proměnné

- Assembler nemá datové typy !!
- Typ proměnné se určuje tím, jak jej čteme či reprezentujeme
- Vše je void* ⇒ všechny proměnné jsou ukazatelé
- Žádná kontrola typů, všechno si řešíte sami, zodpovědnost na programátorovi

Globální proměnné

Velikost		Segment			Registry				
bity	Bajty	.bss	.data	.text					
8	1	resb	db	byte	AH	AL	BH	BL	...
16	2	resw	dw	word	AX	BX	CX	DX	
32	4	resd	dd	dword	EAX	EBX	ECX	EDX	
64	8	resq	dq	qword	???				

Inicializace paměti - deklarace proměnných

```
section .data
    A db 10    ; 8b     char  A = 10;
    B dw 20    ; 16b    short B = 20;
    C dd 30    ; 32b    long  C = 30;

section .bss
    N resb 1    ; 8b
    M resw 1    ; 16b
    O resd 1    ; 32b
```

;v ISU-HUB Watches

Adresování proměnné

Proměnné adresujeme pomocí identifikátoru a hranatých závorek:

```
section .data
    a dd 10

section .text
    mov eax, edx
    mov eax, [a]
    mov [a], eax
    mov [a], [b]
    ; Sběrnice NEDOKÁŽE adresovat více proměnných současně
```

Adresování proměnné - velikost

```
section .data
    a db 0

section .text
    mov al, 100
    mov ax, 100
    mov eax, 100
    mov dl, eax    ; odlišné velikosti !!
    mov [a], 100   ; neznámá velikost !!
```

Adresování proměnné - velikost

Velikost		Segment			Registry
bity	Bajty	.bss	.data	.text	
8	1	resb	db	byte	AH AL BH BL ...
16	2	resw	dw	word	AX BX CX DX
32	4	resd	dd	dword	EAX EBX ECX EDX
64	8	resq	dq	qword	???

(pseudo-instrukce)

Adresování proměnné - velikost

```
section .data
    a db 0
    b dw 0
    c dd 0

section .text
    mov byte [a], 100
    mov word [b], 100
    mov double [c], 100

    mov byte [c], 100   ; ??
    mov [c], byte 100   ; ??
```

Ukázka

- `cv3_0.asm`

Úkoly

- `cv3_1.asm`
- `cv3_2.asm`

Pole

Inicializace pole

```
section .data
    arrA db 10, 20, 30, 40, 50    ; 5x 8b
    arrB dw 10, 20, 30, 40, 50    ; 5x 16b
    arrC dd 10, 20, 30, 40, 50    ; 5x 32b
    stri db “Hello world”, 0      ; 11+1x 8b

section .bss
    arrN resb 5    ; 5x 8b
    arrM resw 5    ; 5x 16b
    arrO resd 5    ; 5x 32b
                                  ; v ISU-HUB nastavujeme Counter
```

Adresování pole

- K adresování položek v poli potřebujeme

  identifikátor a **offset** (posuv)

  [arrA + N]

- Velikost **offset** vždy v *bytech*, tedy posun o 1,2,4 byte.
- První položka je shodná s adresou, tedy posun o 0 byte.

Adresování pole

```
section .data
    arrA db 10, 20, 30    ; 3x 8b
    arrB dw 10, 20, 30    ; 3x 16b

section .text
    mov al, [arrA + 0]    ; 0-tý index 8bit čísla
    mov bl, [arrA + 1]    ; 1-tý index 8bit čísla
    mov cl, [arrA + 2]    ; 2-tý index 8bit čísla

    mov ax, [arrB + 2*0]  ; 0-tý index 16bit čísla
    mov bx, [arrB + 2*1]  ; 1-tý index 16bit čísla
    mov cx, [arrB + 2*2]  ; 2-tý index 16bit čísla
```

Ukázka

- `cv3_3.asm`

Úkol

- `cv3_4.asm`

Indexové registry

Textové řetězce

- 2 vyčleněné registry
  - **ESI** (Source Index) - kde číst
  - **EDI** (Destination Index) - kam zapisovat
- Řetězec je posloupnost **8b** znaků
- Vždy musí být zakončený znakem **0**
- V deklaraci ohraničujeme řetězec uvozovky
- Speciální znaky zadáváme jako hodnotu z ASCII table

Textové řetězce

- Vypsání řetězce funkcí **WriteStringASCIIZ**
  - Vypíše řetězec z adresy uložené v **ESI**
  - Předáváme adresy, tedy žádné závorky
  - Vypisuje dokud nenarazí na **0**

- Načtení řetězce funkcí **ReadString**
  - Funkce načte **EBX** znaků na adresu uloženou v **EDI** a počet úspěšně načtených znaků do **EAX**

Textové řetězce

```
section .data
    retez db “Aloha people!”, 10, 0   ; 14 znaků + \0

section .bss
    vstup resb 6          ; 5 znaků + 0

section .text
main:
    mov esi, retez        ; zdrojová adresa
    call WriteStringASCIIZ

    mov edi, vstup        ; cílová adresa
    mov ebx, 5            ; max počet načtených znaků
    call ReadStringNewLine
    ret
```

Úkol

- `cv3_5.asm`