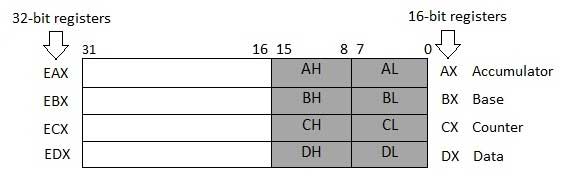
**ASSEMBLER S 32-BITOVYMI REGISTRAMI**

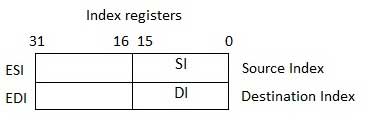


**Dátové registre - ukladáme tam dáta**

EXX - 32 bitové registre (4 bajtové) - EAX,EBX,ECX,EDX

XX - 16 bitové registre (2 bajtové) - AX,BX,CX,DX

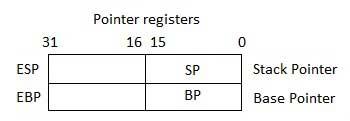
**Indexové registre - používame ich na ukladanie dát a indexovanie**



ESI a EDI - majú veľkosť 32 bitov (4 bajty)

SI a DI - majú veľkosť 16 bitov (2 bajty)

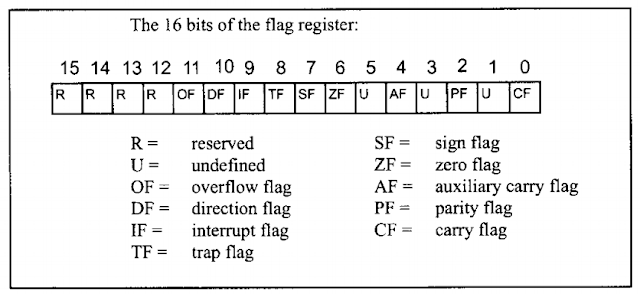
**Ukazovacie registre - ukazujú na nejakú adresu (ja som ich vôbec nevyužíval)**



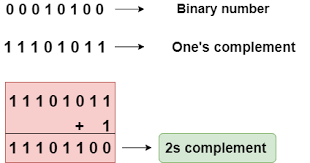
EIP,EBP a ESP - majú veľkosť 32 bitov (4 bajty)

IP,BP a SP - majú veľkosť 16 bitov (2 bajty)

**Flags**



**B1/B2 komplement**



**Definovanie premenných a polí:**

premenné definujeme v tvare názov\_premennej veľkosť obsah

napr.

pom DB ?; premenná pom typu DB (1 bajt) ktorá nemá žiadnu hodnotu

retazec DW "ahoj",0 ; retazec retazec typu DW (každý znak má veľkosť 2 bajty) ktorý obsahuje "ahoj" a je ukončený(uzavretý) nula na konci to je dôležité aby keď definuješ reťazec čísel alebo znakov ho uzavrieš inak ti bude pretekať do ďalšieho riadku

retazec2 DD 3 dup (?) ;retazec retazec 2 ktorý má veľkosť DD (každá pozícia má veľkosť 4 bajty) a obsahuje 3 prázdne polia kde nieje nič

keď chceme aby retazec2 obsahoval na každej pozicii rovnaký znak dáme ho do zátvorky

retazec2 DD 3 dup (a) ; retazec2 obsahuje "aaa"

**Základné príkazy:**

**mov** - príkaz na presun dát

napr. mov al,bl - z registra bl presunie dáta do registra al (dáta ostanú aj v bl)

**inc** - príkaz ktorý zvýši register alebo premennú

napr. inc esi - zvýši obsah registra esi o 1 ak v esi bude napr. znak A tak po vykonaní tam bude znak B

**dec** - príkaz ktorý zníži register alebo premennú

napr. dec esi - zvýši obsah registra esi o 1 ak v esi bude napr. znak C tak po vykonaní tam bude znak B

**cmp** - príkaz compare ktorý porovnáva 2 rovnako veľké premenne či registre či pod. a na základe porovnania vyhodnotí podmienku

napr. cmp eax,edx

je sem

sem:

...

**jne** - jump not equal - porovnáva či sa nerovnajú členy ak je to pravda tak skočí na dané návestie

**jg** - jump greater - porovnáva či je prvý člen väčší ako druhý ak je pravda skočí na dané návestie

**jl** - jump less - porovnáva či je prvý člen menší ako druhý ak je pravda skočí na dané návestie

**jge** - jump greater or equal - porovnáva či je prvý člen vačší alebo rovný druhému ak je pravda skáče

**jle** - jump greater or equal – porovnáv8a či je prvý člen menší alebo rovný druhému ak je pravda skáče

**Zásobník**

-využíva príkazy **push** a **pop**

teda napr. keď chceme uložiť obsah registra EAX do zásobníka tak napíšeme push eax a keď ho chceme vybrať a vložiť napr. do ebx napíšeme pop ebx;

pushovat a popovat môžeme len 4 bajtové (32-bitové registre) - EAX,EBX,ECX,EDX,ESI,EDI,EBP,ESP,EIP a v programe alebo procedúre v ktorej si uložíme do zásobníka dačo si musíme všetky tieto uložené údaje potom vybrať inak ti bude vyhadzovať error tabuľka

**Aritmetické operácie:**

**sčítanie - príkaz add**

mov ax,7

mov bx,3

add ax,bx

do ax sme si uložili 7 do bx 3 add ax,bx pripočíta k ľavému registru alebo premennej ten pravý a výsledok sa uloží do ax (do ľavej premennej) teda bude tam 10

**odčítanie - príkaz sub**

mov ax,8

mov bx,3

sub ax,bx

do ax sme si uložili 8 do bx 3 sub ax,bx odčíta od ľavého registra (premennej) ten pravý a výsledok sa uloží do ax (do ľavej premennej) teda bude tam 5

**násobenie - príkaz mul**

mov ax,8

mov bx,3

mul bx

do ax sme si uložili 8 do bx 3 mul bx vynásobí register ax s registrom bx a výsledok sa uloží do ax teda bude tam 24

**delenie - príkaz div -** robí celočíselný podiel a zvyšok uloží do iného registra

delenie pomocou 2 bajtových registrov:

mov dx,0

mov ax,8

mov bx,3

div bx

v dx je uložený vždy zvyšok po delení 2 bajtových čísel treba vždy tento register vynulovať lebo ak nebude nulový a bude sa robiť div tak bude vyhadzovať error tabuľka takže vždy treba napísať mov dx,0 - znuluje dx

ax je prvé číslo ktorým sa bude deliť ďalšie

bx je druhé číslo ktorým sa bude deliť prvé číslo

div bx delí ax/bx = 8/3 výsledok je 2 ktorý sa uloží do ax, zvyšok je 2 ten sa uloží do dx

delenie pomocou 1 bajtových registrov:

mov ah,0

mov al,10

mov bl,3

div bl

v ah je uložený vždy zvyšok po delení 1 bajtových čísel treba vždy tento register vynulovať lebo ak nebude nulový a bude sa robiť div tak bude vyhadzovať error tabuľka takže vždy treba napísať mov ah,0 - znuluje ah

al je prvé číslo ktorým sa bude deliť ďalšie

bl je druhé číslo ktorým sa bude deliť prvé číslo

div bl delí al/bl = 10/3 výsledok je 3 ktorý sa uloží do al, zvyšok je 1 ten sa uloží do ah

ROZDIEL MEDZI 1 bajtovými a 2 bajtovými číslami?

1 bajtové neznamienkové majú rozsah od 0 do 255 (28-1 (keďže tam berieme aj 0, preto -1)

1 bajtové znamienkové majú rozsah od -128 do 127 (platí to iste v zátvorke 28 čísel polovica - a polovica +, 0 je nezáporné rata sa do kladných)

2 bajtové neznamienkové majú rozsah od 0 do 65535 (216-1 (keďže tam berieme aj 0, preto -1)

2 bajtové znamienkové majú rozsah od -32768 do 32767 (platí to iste v zátvorke 216 čísel polovica - a polovica +, 0 je nezáporné rata sa do kladných)