ARHITECTURA SISTEMELOR DE CALCUL
– Seminar 3 –

1. Instrucțiuni de comparare
2. Salturi
3. Instrucțiuni de ciclare
4. Șiruri

1. INSTRUCȚIUNI DE COMPARARE

CMP

Sintaxa: `cmp d, s`

Efect: execută **operația aritmetică `d - s`** (fără să depună rezultatul în `d`) și modifică flag-urile în funcție de rezultatul operației efectuate

Flag-uri afectate: OF, SF, ZF, AF, PF, CF

Constrângeri:

- operatorul destinație `d` poate fi un registru sau o locație în memorie;
- operandul sursă `s` poate fi un registru, o locație în memorie sau o valoare imediată (constantă);
- operanzii trebuie să aibă aceeași dimensiune de reprezentare (octet/cuvânt/dublucuvânt);
- operanzii nu pot să fie SIMULTAN locații de memorie (unul dintre ei trebuie să fie un registru).

Observații

Deşi numele acestei instrucțiuni este CMP este important de subliniat că, în realitate, această instrucțiune NU COMPARĂ nimic, nestabilind niciun criteriu de comparaţie şi neluând de fapt nicio decizie. Ea doar PREGĂTEȘTE decizia corespunzător cu flagurile setate, comparaţia efectivă și decizia corespunzătoare fiind luată concret de instrucţiunea de salt condiționat care va fi folosită ulterior instructiunii CMP !

Dacă nu folosim ulterior nicio instrucțiune de salt condiționat, CMP nu are nici un rol concret în vreo comparație, ea reprezintă doar o simplă scădere fictivă cu rol de afectare a flagurilor şi nu îşi va merita în niciun caz numele de CMP (compare).

TEST

Sintaxa: `test d, s`

Efect: execută **operația logică `d AND s`** (fără să depună rezultatul în `d`) și modifică flag-urile în funcție de rezultatul operației efectuate

Flag-uri afectate: poate modifica doar SF, ZF, PF (CF = 0, OF = 0, AF - nedefinit)

Constrângeri: identice cu cele menționate pentru instrucțiunea CMP

Observații

Deşi numele acestei instrucțiuni este TEST este important de subliniat ca în realitate această instrucțiune NU TESTEAZA nimic, nestabilind niciun criteriu de testare şi neluând de fapt nicio decizie, ci ea doar PREGĂTEȘTE decizia corespunzător cu flagurile setate, criteriul de testare, testarea efectivă și decizia corespunzătoare fiind luată concret de instrucţiunea de salt condiționat care va fi folosită ulterior instrucțiunii TEST !

Dacă nu folosim ulterior nicio instrucțiune de salt condiționat, TEST nu are nici un rol concret în vreo testare, ea reprezentând doar o simplă operație AND (bit cu bit) cu rol de afectare a flagurilor şi nu îşi va merita în niciun caz numele de TEST (testarea unei condiții).

2. SALTURI

Salturile pot fi de 2 (două) tipuri: *necondiționate* sau *condiționate*.

2.a. Saltul necondiționat

Există o singură instrucțiune de salt necondiționat:

JMP

Sintaxa: `jmp op`

unde `op` poate fi o etichetă, un registru sau o locație memorie ce conține o adresă.

Efect: execuția va continua cu instrucțiunea care urmează după etichetă, care se află la adresa conținută în registru sau la adresa conținută în locația de memorie

Instrucțiunea JMP poate fi utilizată pentru a executa oricare dintre următoarele 3 (trei) tipuri de salturi:

- *short jump*: un salt la o instrucțiune al cărei offset este cuprins între [−128, 127] octeți față de valoarea curentă din registrul EIP
- *near jump* (intrasegment jump): un salt la o instrucțiune aflată în segmentul de cod curent (referit de către registrul de segment CS)
- *far jump* (intersegment jump): un salt la o instrucțiune aflată într-un alt segment de cod

2.b. Instrucțiuni de salt condiționat

Jcc (Conditional jump instructions)

Sintaxa: `jcc eticheta`

Efect: transferă controlul programului instrucțiunii care urmează după `eticheta`, dacă condiția specificată de `cc` (*condition code* associated with de instruction) este adevărată

Lista completă a instrucțiunilor de salt condiționat este prezentată în *"Intel 64 and IA-32 Architectures Software Developer's Manual, volume 1"*, care poate fi consultat la această adresă (*vezi secțiunea 7.3.8*):

`https://cdrdv2.intel.com/v1/dl/getContent/671436`

Mnemonica	Semnificație	Condiția verificată
JB **JNAE** **JC**	este inferior nu este superior sau egal există transport	CF=1
JAE **JNB** **JNC**	este superior sau egal nu este inferior nu există transport	CF=0
JBE **JNA**	este inferior sau egal nu este superior	CF=1 sau ZF=1
JA **JNBE**	este superior nu este inferior sau egal	CF=0 şi ZF=0
JE **JZ**	este egal este zero	ZF=1
JNE **JNZ**	nu este egal nu este zero	ZF=0
JL **JNGE**	este mai mic decât nu este mai mare sau egal	SF≠OF
JGE **JNL**	este mai mare sau egal nu este mai mic decât	SF=OF
JLE **JNG**	este mai mic sau egal nu este mai mare decât	ZF=1 sau SF≠OF
JG **JNLE**	este mai mare decât nu este mai mic sau egal	ZF=0 şi SF=OF
JP **JPE**	are număr par de biți 1 în octetul inferior al UOE	PF=1

JNP **JPO**	are număr impar de biți 1 în octetul inferior al UOE	PF=0
JS	are semn negativ	SF=1
JNS	nu are semn negativ	SF=0
JO	există depășire	OF=1
JNO	nu există depășire	OF=0

Compararea a două numere poate fi făcută cu semn sau fără semn. De aceea, convenim că:

– atunci când se compară două numere fără semn se folosesc termenii ***"below"*** (inferior, sub), respectiv ***"above"*** (superior, deasupra, peste);

– atunci când se compară două numere cu semn se folosesc termenii ***"less then"*** (mai mic decât) și ***"greater then"*** (mai mare decât).

Pentru a facilita alegerea corectă de către programator a instrucțiunilor de salt condiționat în raport cu rezultatul unei comparații (cu semn sau fără semn), dăm următorul tabel:

Relația dintre operanzi	Comparație cu semn	Comparație fără semn
`d = s`	`JE`	`JE`
`d ≠ s`	`JNE`	`JNE`
`d < s`	`JL`	`JB`
`d ≤ s`	`JLE`	`JBE`
`d > s`	`JG`	`JA`
`d ≥ s`	`JGE`	`JAE`

Observații

– nu instrucțiunea CMP este cea care face distincție între o comparație cu semn și una fără semn;

– rolul de a interpreta în mod diferit (cu semn sau fără semn) rezultatul final al comparației revine EXCLUSIV instrucțiunilor de salt condiționat care urmează după instrucțiunea CMP;

– dacă considerăm cazul `s = 0`, tabelul rămâne valabil.

3. INSTRUCȚIUNI DE CICLARE

LOOP

Sintaxa: `loop eticheta`

Condiția testată: ECX = 0

Efect: decrementează conținutul registrului ECX, apoi, dacă condiția nu este adevărată, execută salt la instrucțiunea care urmează după `eticheta`

LOOPE (LOOP while Equal)

Sintaxa: `loope eticheta`

Condiții testate: ECX = 0 sau ZF = 0

Efect: decrementează valoarea registrului ECX, apoi, dacă niciuna din condiții nu e adevărată, execută salt la instrucțiunea care urmează după `eticheta`

LOOPNE (LOOP while Not Equal)

Sintaxa: `loopne eticheta`

Condiția de terminare: ECX = 0 sau ZF = 1

Efect: decrementează valoarea registrului ECX, apoi, dacă niciuna dintre condiții nu e adevărată, execută salt la instrucțiunea care urmează după `eticheta`

4. ȘIRURI

4.a. Șiruri numerice (de octeți/cuvinte/dublucuvinte/quadwords)

Reprezentarea datelor în memorie (little endian)

```
s1 db 12, 13, 14
```

s1

0c	0d	0e							

```
s2 dw 1234h, 5678h
```

s2

34	12	78	56						

```
s3 dd 12345678h, 1234abcdh
```

s3

78	56	34	12	cd	ab	34	12		

4.b. Șiruri de caractere

Caracterele sunt reprezentate în cod ASCII (American Standard Code for Information Interchange).

Reprezentarea datelor în memorie (little endian)

```
s1 db 'a', 'b', 'c'
```

s1

61	62	63							

```
s2 dw 'a', 'b', 'c'
```

s2

61	00	62	00	63	00				

```
s3 dd 'a', 'b'
```

s3

61	00	00	00	62	00	00	00		

4.c. Determinarea lungimii unui șir

Determinarea lungimii unui șir (în OCTEȚI) se poate face folosind contorul de locații.

Contorul de locații este un număr întreg gestionat de către asamblor. În orice moment, valoarea acestui contor coincide cu numărul de octeți generați corespunzător instrucţiunilor şi directivelor deja întâlnite în cadrul segmentului respectiv (deplasamentul curent în cadrul segmentului).

Programatorul poate să acceseze (poate să citească, dar nu poate să modifice) valoarea curentă a contorului de locații utilizând simbolul `$`.

Exemplu

Fie segmentul de date care urmează:

```
...
; segmentul de date
segment data use32 class=data
  s db 'a', 'b', 'c', 'd', 'e'
  len equ $-s                    ; lungimea șirului s (în octeți)

; segmentul de date
segment code use32 class=code
    start:
       ...
```

EXERCIȚII

1. Se dă un șir de caractere S. Se cere șirul de caractere D obținut prin copierea șirului S.

2. Se dă un șir de caractere S format din litere mici. Să se construiască un șir de caractere D care să conțină literele din șirul inițial transformate în majuscule.

3. Se dă un șir de numere întregi S. Să se construiască șirul P care conține toate numerele pare din S și șirul R care conține toate numerele impare din S.

4. Se dă un șir de cuvinte S. Să se determine suma numerele formate din biții 6-9 ai fiecărui cuvânt din șirul S.

5. Se dă un șir de cuvinte S. Să se formeze șirul de octeți D care conține octeții superiori rotiți spre stânga cu valoarea octeților inferiori din fiecare cuvânt al șirului de cuvinte S.