Rechnerarchitektur

Labor 8

Organisatorische Aspekte

- Test 2 -> Woche 9 (24.-25. 11.2020)
- Inhalt: Labor 6,7,8
- 3 Aufgaben
- Anforderung: institutionelle E-mail Adresse (outlook)

Wiederholung

- Befehle, welche Flags setzen
- Sprungbefehle
- bedingte Sprungbefehle
- Zeichenfolgen
- Offset
- Positionszähler ($)

Stringbefehle

- haben implizite Operanden

- Arten von Stringbefehle:
  - die eine Quellfolge und eine Zielfolge verwenden (MOVSB, MOVSW, MOVSD, CMPSB, CMPSW, CMPSD)
  - die nur eine Quellfolge verwenden (LODSB, LODSW, LODSD)
  - die nur eine Zielfolge verwenden (STOSB, STOSW, STOSD, SCASB, SCASW, SCASD)

- Move String = MOVS
- Compare String = CMPS
- Load String = LODS
- Store String = STOS
- Scan String = SCAS

- Die Stringbefehle sind gekennzeichnet durch:
  - Die Art der Elemente (Bytes, Wörter oder Doppelwörter) => wird durch den letzten Buchstaben der verwendeten Anweisung angegeben (B = Byte, W = Wort, D = Doppelwort), wobei beide Zeichenfolgen denselben Typ haben
  - Die Adressierung erfolgt immer über die Registerpaare DS:ESI und (oder) ES:EDI
    - in DS: ESI - für die Quellstring
    - in ES: EDI - für die Zielstring
  - Richtung der Durchlauf => wird durch den Wert im DF-Flag angezeigt (0 - von kleineren Adressen zu größeren Adressen, 1 - von größeren Adressen zu kleineren Adressen.)
  - Die Anzahl der Elemente => bei Bedarf, in CX oder ECX setzen

	DF = 0	**DF = 1**
Byteoperation	+1	-1
Wortoperation	+2	-2
Doppelwortoperation	+4	-4

Stringbefehle zur Datenübertragung: LODS_ , STOS_ , MOVS_

LODS_

(LODSB, LODSW, LODSD)

- schreibt ein Byte/Wort/Doppelwort von der Adresse<DS:ESI> nach AL/AX/EAX ab
- DF=0 -> ESI mit Anzahl von gelesenen Bytes inkrementiert
- DF=1 -> ESI mit Anzahl von gelesenen Bytes dekrementiert
- Beispiel:
    - bsp DB ‘Das ist ein Beispiel!’
    - ...
    - MOV ESI, bsp ; DS:ESI zeigt auf bsp
    - CLD ; DF=0 -> Inkrement
    - LODSB. ; AL -> ‘D’

LODSB	In AL se incarcă octetul de la adresa <DS:ESI> Daca DF=0 atunci inc(ESI), altfel dec(ESI)
LODSW	In AX se incarcă cuvantul de la adresa <DS:ESI> Daca DF=0 atunci ESI:=ESI+2, altfel ESI:=ESI-2
LODSD	In EAX se incarcă dublucuvantul de la adresa <DS:ESI> Daca DF=0 atunci ESI:=ESI+4, altfel ESI:=ESI-4

STOS_

(STOSB, STOSW, STOSD)

- speichert das Byte/Wort/Doppelwort in AL/AX/EAX nach das Byte/Wort/Doppelwort von der Adresse<ES:EDI>
- DF=0 -> EDI mit Anzahl von gelesenen Bytes inkrementiert
- DF=1 -> EDI mit Anzahl von gelesenen Bytes dekrementiert

STOSB	La adresa <ES:EDI> se incarcă octetul din AL Daca DF=0 atunci inc(EDI), altfel dec(EDI)
STOSW	La adresa <ES:EDI> se incarcă cuvantul din AX Daca DF=0 atunci EDI:= EDI+2, altfel EDI:= EDI-2
STOSD	La adresa <ES:EDI> se incarcă dublucuvantul din EAX Daca DF=0 atunci EDI:= EDI+4, altfel EDI:= EDI-4

MOVS_

(MOVSB, MOVSW, MOSD)

- schreibt ein Byte/Wort/Doppelwort von der Adresse<DS:ESI> nach <ES:EDI> ab
- DF=0 -> ESI + EDI mit Anzahl von gelesenen Bytes inkrementiert
- DF=1 -> ESI + EDI mit Anzahl von gelesenen Bytes dekrementiert

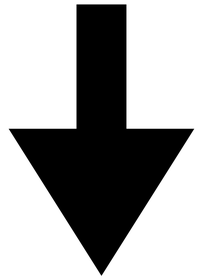
MOVSB	La adresa <ES:EDI> se incarca octetul de la adresa <DS:ESI> Daca DF=0 atunci inc(SI), inc(DI), altfel dec(SI), dec(DI)
MOVSW	La adresa <ES:EDI> se incarca cuvantul de la adresa <DS:ESI> Daca DF=0 atunci ESI:= ESI+2, EDI:= EDI+2, altfel ESI:= ESI-2, EDI:= EDI-2
MOVSD	La adresa <ES:EDI> se incarca dublucuvantul de la adresa <DS:ESI> Daca DF=0 atunci ESI:= ESI+4, EDI:= EDI+4, altfel ESI:= ESI-4, EDI:= EDI-4

Beispiel

- ; Sei eine Quellzeichenfolge (Wörter). Schreibt diese folge in einen Zielfolge ab. Die Anzahl der Elemente ist gegeben

```
mov ECX, dim_sir     ; Anzahl der Elemente in der String

mov ESI, sir_source  ; Offset des sir_source in ESI speichern

mov EDI, sir_dest.   ; Offset des sir_dest in EDI speichern

CLD

Again:

        LODSW

        STOSW

LOOP Again
```

- **LODS** + **STOS** = **MOVS**



```
Again:

        MOVSW

LOOP Again
```

Stringbefehle zur Datenzugriff und Vergleich: SCAS_ , CMPS_

SCAS_

(SCASB, SCASW, SCASD)

- Vergleicht das Byte/Wort/Doppelwort in AL/AX/EAX mit dem in <ES:EDI> und setzt die Flags wie bei CMP *(CMP AL/AX/EAX, <ES:EDI>)*
- DF=0 -> EDI mit Anzahl von gelesenen Bytes inkrementiert
- DF=1 -> EDI mit Anzahl von gelesenen Bytes dekrementiert

SCASB	CMP AL, <ES:EDI> Daca DF=0 atunci inc(EDI), altfel dec(EDI)
SCASW	CMP AX, <ES:EDI> Daca DF=0 atunci EDI:= EDI+2, altfel EDI:= EDI-2
SCASD	CMP EAX, <ES:EDI> Daca DF=0 atunci EDI:= EDI+4, altfel EDI:= EDI-4

CMPS_

(CMPSB, CMPSW, CMPSD)

- Vergleicht das Byte/Wort/Doppelwort in <ES:ESI> mit dem in <ES:EDI> und setzt die Flags wie bei CMP *(CMP <ES:ESI>, <ES:EDI>)*
- DF=0 -> ESI + EDI mit Anzahl von gelesenen Bytes inkrementiert
- DF=1 -> ESI + EDI mit Anzahl von gelesenen Bytes dekrementiert

CMPSB	CMP <DS:ESI>, <ES:EDI> Daca DF=0 atunci inc(ESI), inc(EDI), altfel dec(ESI), dec(EDI)
CMPSW	CMP <DS:ESI>, <ES:EDI> Daca DF=0 atunci ESI:= ESI+2, EDI:= EDI+2, altfel ESI:= ESI-2, EDI:= EDI-2
CMPSD	CMP <DS:ESI>, <ES:EDI> Daca DF=0 atunci ESI:= ESI+4, EDI:= EDI+4, altfel ESI:= ESI-4, EDI:= EDI-4

Beispiel

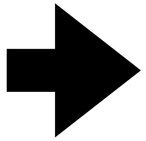
```
; Geben sei eine Zeichenfolge von Bytes. Sucht das letzte Zeichen “m".
    MOV EDI, sir+l-1
    MOV AL, ‘m’
    MOV ECX, l
    STD
    Suche:
        SCASB
        JE Gefunden
    LOOP Suche
    ...
    Gefunden:
        INC EDI    ; kehrt zu dem vorigen Charakter zurück, bevor man EDI  dekrementiert wurde
```

Wiederholungspräfixe für Stringbefehle

Again:

Stringbefehl

LOOP Again



wiederholungsPräfix Stringbefehl

- **REP** - nach jeder Ausführung des Stringbefehls -> EXC decrementiert (- -) und falls ECX != 0 führt den Stringbefehl wieder
- **REPE** *(Repeat While Equal)* und **REPZ** *(Repeat While Zero)* -
    - führt die gleiche Schritte wie REP
    - zweites Abbruchkriterium: Ungleichheit der Operanden (-> ZF=0)
- **REPNE** *(Repeat While Not Equal) und* **REPNZ** *(Repeat While Not Zero)*
    - führt die gleiche Schritte wie REP
    - zweites Abbruchkriterium: Gleichheit der Operanden (-> ZF=1)

Bemerkungen

- die Wiederholungspräfixe können **NUR** mit den SCAS_ und CMPS_ - Stringbefehlen verwendet werden.
- LODS_, STOS_, MOVS_ setzen keine Flags
- SCAS_ und CMPS_ setzen die Flags, weil diese den CMP ausführen