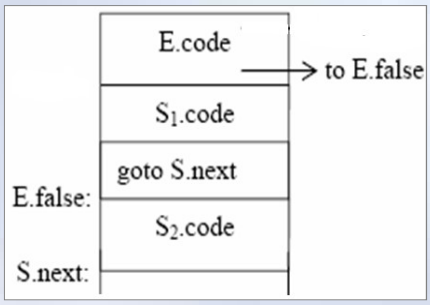
Cấu trúc điều khiển của ngôn ngữ bậc cao

- Cấu trúc điều khiển của ngôn ngữ bậc cao
  - Cấu trúc rẽ nhánh **IF ELSE**
  - Biểu thức phức hợp cùng với **AND**, **OR**
  - Cấu trúc lặp **WHILE**
- Chuyển đổi sang hợp ngữ theo nguyên tắc sinh mã trong CTD

Cấu trúc rẽ nhánh IF .....ELSE



If E Then S1 Else S2

```
if( Op1 == Op2 )
   X = 1;
else
   X = 2;
```

```
     MOV EAX, op1
     CMP EAX, Op2
     JNE L1
     MOV X, 1
     JMP L2
L1:  MOV X, 2
L2:
```

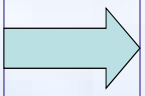
Cấu trúc rẽ nhánh IF ELSE→Số không dấu

```
if( EBX <= ECX ){
    EAX = 5
    EDX = 10
}
```

```
    CMP EBX, ECX
    JA Next    ; JNBE NEXT
    MOV EAX, 5
    MOV EDX, 10
Next:
```


Cấu trúc rẽ nhánh IF ELSE→Số có dấu

```
if( var1 <= var2 )
    var3 = 10;
else
    var3 = 6;
```



```
    MOV EAX, Var1
    CMP EAX, Var2
    JLE L1          ;JNG L1
    MOV Var3, 6
    JMP L2
L1: MOV Var3, 10
L2:
```

```
    MOV EAX, Var1
    CMP EAX, Var2
    JNLE L1         ;JG L1
    MOV Var3, 10
    JMP Next
L1: MOV Var3, 6
Next:
```

Biểu thức với toán tử AND

- Nếu biểu thức đầu tiên có giá trị **sai**, biểu thức tiếp theo có thể bị bỏ qua;
- **Ví dụ:** if (AX > BX) && (BX >CX) X= 1

```
        CMP AX, BX; biểu thức 1
        JA L1
        JMP Next
L1:     CMP BX, CX; biểu thức 2
        JA L2
        JMP Next
L2:     MOV X, 1    ; Cả 2 thỏa mãn
Next:
```

```
CMP AX, BX
JNA Next
CMP BX, CX
JNA NEXT
MOV X, 1
Next:
```


Biểu thức với toán tử AND

```
if( ebx <= ecx  && ecx > edx ){ //Số không dấu
    eax = 5;
   edx = 6;
}
```

```
        CMP EBX, ECX
        JA NEXT
        CMP ECX, EDX
        JBE Next
        MOV EAX, 5
        MOV EDX, 6
Next:
```

Biểu thức với toán tử OR

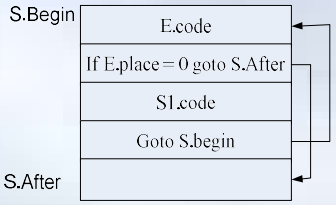
- Nếu biểu thức đầu tiên có giá trị **đúng**, biểu thức tiếp theo có thể bị bỏ qua;
- **Ví dụ:** if (AX > BX) || (BX >CX) X= 1

```
        CMP AX, BX; biểu thức 1
        JA L1
        CMP BX, CX; biểu thức 2
        JBE Next; Cả 2 không thỏa mãn
L1:     MOV X, 1
Next:
```


Cấu trúc lặp WHILE

Bao gồm

- Biểu thức điều kiện IF
- Thân vòng lặp
- Nhảy không điều kiện tới biểu thức điều kiện đầu



Ví dụ while (eax <= edx) eax=eax + 1

```
Top :   CMP EAX, EDX
        JA Next
        INC EAX
        JMP TOP
Next:
```

Nội dung chính

- Thanh ghi trạng thái processor
- Các lệnh sao chép dữ liệu
- Các lệnh số học và logic
- Các lệnh thao tác với bit
- Các lệnh điều khiển
- **Các câu lệnh với chuỗi**
- Các câu lệnh khác

Câu lệnh

- X86 cung cấp một số lệnh liên quan chuỗi byte, word, dword
  - Lệnh: Movs_, Stos_, Lods_, Cmps_, Scas_,…
- Chỉ ra kích thước dữ liệu bởi hậu tố [b, w, d]
  - Kiểu dword có từ 386
- Thường dùng kèm các chỉ thị lặp REP, REPZ..
  - CX chứa số lần lặp, sẽ giảm tự động đi 1 đơn vị
- Thanh ghi SI, DI thay đổi phụ thuộc cờ hướng
  - Thay đổi cờ hướng: CLD, STD

Câu lệnh **MOVSB, MOVSW, MOVSD**

[REP] **MOVSB/ MOVSW/ MOVSD**

- Copy dữ liệu từ vị trí bộ nhớ được xác định bởi **DS:SI** sang vị trí trong **ES:DI**
  - Thực hiện: ES:[DI] ←DS:[SI]
  - **SI**, **DI** được tăng /giảm tự động 1/2/4 byte
    - Tăng khi DF=0=UP; Giảm khi DF=1=DOWN
- Đặt chỉ thị **REP** trước câu lệnh
  - CX xác định **số lần** lặp. Kết thúc lặp CX = 0

Ví dụ: Xóa phần tử đầu tiên của vector

```
.model tiny    ;
.data
  Vec DB "HHello world$"
  len = $-Vec  ;13
.code
   .startup
   LEA SI,Vec+1
   LEA DI, Vec
   Mov CX, len -1   ;12
   CLD
   REP MOVSB
.exit
end
```

Ví dụ: Chuyển 512Byte từ vị trí 7C00 về 6000

```
.code
   .startup
      XOR AX, AX
      MOV DS, AX
      MOV ES, AX
      MOV SI, 7C00h          ; DS : SI => 7C00h
      MOV DI, 6000h          ; ES : DI => 6000h
      MOV CX, 512
      CLD
      REP MOVSB
      JMP 0: Next  ;Dat lai con tro lenh vao vi tri moi
Next:
end
```


Câu lệnh **so sánh xâu**

[REP_] **CMPSB/ CMPSW/ CMPSD**

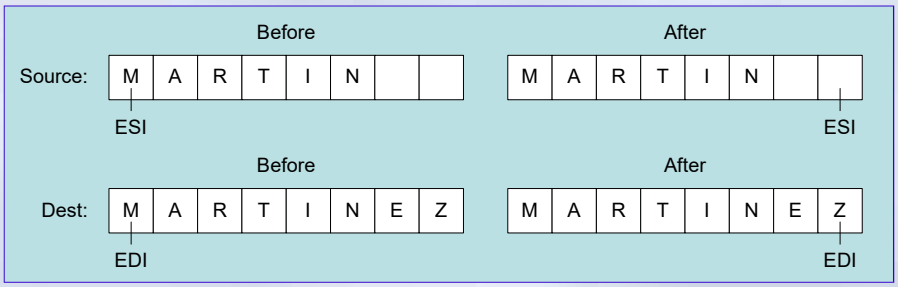
- Cập nhật thanh ghi cờ theo kết quả của phép so sánh giá trị tại DS:[SI] và ES:[DI]
  - **SI**, **DI** được tăng /giảm tự động 1/2/4 byte
- REPZ/REPE:
  - Lặp lại khi CX >0 và 2 giá trị so sánh bằng nhau
- REPNZ/REPNE
  - Lặp lại khi CX >0 và 2 giá trị không bằng nhau
- Có thể dùng lệnh nhảy có điều kiện sau đó

Câu lệnh **so sánh xâu**

Giá trị các thanh ghi SI, DI sau khi thực hiện

CLD

REPE CMPSB



Câu lệnh **SCA**n **S**tring

[REP_] **SCASB/ SCASW/ SCASD**

- Cập nhật thanh ghi cờ theo kết quả của phép so sánh giá trị AL/AX/EAX với ES:[DI]
  - **DI** được tăng /giảm tự động 1/2/4 byte
- **REPZ/REPE**:
  - Lặp lại khi CX >0 và 2 giá trị còn bằng nhau
- **REPNZ/REPNE**
  - Lặp lại khi CX >0 và 2 giá trị không bằng nhau
- **CX**: Số lần lặp tối đa
  - Kết thúc lặp, CX >0 → đ/k lặp không thỏa mãn

Câu lệnh **SCA**n **S**tring : Tìm số không đầu tiên

```
.data
   Vec DW 34, 50, 24, -57, 22, 0 , 20
   Len =($-Vec)/ TYPE Vec
.code
    .startup
    LEA DI, Vec
    Mov AX, 0
    MOV CX, Len
    REPNE SCASW ;Lặp nếu không bằng
    JNZ notExist;,Cờ zero bị xóa khi ES:[DI] – AL =0
    SUB DI, TYPE Vec
  notExist:
    .exit
end
```


Câu lệnh **STO**re **S**tring

[REP] **STOSB/ STOSW/ STOSD**

- Lưu giá trị AL/AX/EAX với ES:[DI]
  - **DI** được tăng /giảm tự động 1/2/4 byte
- Sử dụng **REP** → **CX**: Số lần lặp
- Ví dụ: Khởi tạo Vector Vec gồm 100 word

```
CLD
MOV CX, 100; 200 byte
LEA DI, Vec
XOR AX, AX; Xóa AX
REP STOSW
```

Câu lệnh **STO**re **S**tring→Ví dụ

```
.model tiny ; Ki?u b? nh?
.386                                ;Processor 32bit
.data
      Vec DD 10000 DUP(?)
      Len1 =($-Vec)/ TYPE Vec
      Len2 = $-Vec
      Msg1 DB "Start reset 10000 DWORD ",13,10,"$"
      Msg2 DB "Start reset 40000 BYTE  ",13,10,"$"
.code
   .startup  ;?i?m b?t ??u c?a ch??ng trình
   Mov AH, 9
   LEA DX, Msg1
   Int 21h
   MOV CX,40000          ;
   MOV EAX,0FFFFFFFFh   ;
   L1:                                ;
    PUSH CX              ;
    LEA DI, Vec      ;
    MOV CX, Len1     ;
    REP STOSD        ;
    POP CX            ; Khôi phục CX
    LOOP L1           ; Lặp lại vòng lặp ngoài
   Mov AH, 9
   LEA DX, Msg2
   Int 21h
   MOV CX,40000          ;
   MOV EAX,0FFFFFFFFh   ;
   L2:                                ;
    PUSH CX              ;
     LEA DI, Vec      ;
    MOV CX, Len2     ;
     REP STOSB        ;
     POP CX           ; Khôi phục CX
    LOOP L2              ;
    .exit             ;
End
```


Câu lệnh **LOD**e **S**tring

[REP] **LODSB/ LODSW/ LODSD**

- Copy giá trị DS:[SI] vào AL/AX/EAX
  - **SI** được tăng /giảm tự động 1/2/4 byte
- Cho phép sử dụng với **REP** nhưng vô nghĩa
  - Mỗi lần lặp, giá trị AL/AX/EAX bị thay đổi→ Sử dụng REP, AL/ AX/ EAX chứa giá trị cuối

Ví dụ → Chuyển thành xâu chữ thường

```
.model tiny    ;
.386           ;Processor 32bit
.data
  Msg DB "HeLLO WoRld !$"
  Len =($-Msg)
.code
   .startup
   LEA DI, Msg
   MOV SI, DI ; SI=DI
   Mov CX, Len
   CLD         ;clear direction
Convert:
      LODSB
      CMP AL, 'A'
      JB NotUpper
      CMP AL, 'Z'
      JA NotUpper
      OR AL, 20h
  notUpper:
      STOSB
      LOOP Convert
      Mov AH, 9
      Mov DX, OFFSET Msg
      Int 21h
   .exit
end
```