Indirect Addressing

Muhammad Afzaal
m.afzaal@nu.edu.pk

Book Chapter

- “Assembly Language for x86 Processors”
- Author “Kip R. Irvine”
- 6th Edition
- Chapter 4
  - Section 4.4
  - Section 4.5

Indirect Operands (1/3)

- In Protected Mode
  - A 32-bit general purpose register can be used as an indirect operand surrounded by square brackets
  - The register contains the address of variable
- In Real-Address Mode
  - A 16-bit register holds the offset of variable
  - Any of SI, DI, BX or BP can be used
- Indirect Operands are useful to step through arrays

Indirect Operands (2/3)

- Protected Mode

```
.data
val DB 10h, 20h, 30h
.code
MOV esi, OFFSET val
MOV al, [esi]
INC esi
MOV bl, [esi]
```

Indirect Operands (3/3)

- Real-Address Mode

```
.data
val DB 10h, 20h, 30h
.code
MOV si, OFFSET val
MOV al, [si]
INC si
MOV bl, [si]
```

Pointer

- A pointer can be declared in the following way

```
.data
val DW 10h
vptr DW val
.code
MOV si, vptr
MOV al, [si]
```

JMP Instruction

- Causes an unconditional transfer to a destination
- Destination is identified by a label which is translated into offset at assemble time
- When CPU executed JMP, the offset of destination is moved into the IP

```
dest:

.

.

JMP dest
```

LOOP Instruction

- Loop instruction creates a counting loop
- Syntax is `LOOP target`
- Logic:
  - ECX ← ECX – 1
  - If ECX != 0, jump to `target`
- Implementation
  - Assembler calculates the distance in bytes between the offset of the following instruction and the offset of the target label. It is called relative offset
  - Relative offset is added to EIP
- How to use ECX in case of nested loops?

Summing an Integer Array

```
.data
arr DB 10h, 20h, 30h, 40h
.code
MOV si, OFFSET arr
MOV ecx, LENGTHOF arr
MOV ax, 0
L1:
    ADD ax, [si]
    ADD si, TYPE arr
    LOOP L1
```

Copying a String

```
.data
src DB “This is source”, 0
dst DB SIZEOF src DUP(0)
.code
MOV si, 0
MOV ecx, SIZEOF src
L1:
    MOV al, src[si]
    MOV dst, al
    INC si
    LOOP L1
```