Org: Assembly Programming

Alex Chi

Update: March 31, 2020

Contents

1 Programming Language **2**

2 Form of a Statement **2**

2.1 Shell of a Real Program 2

2.2 Model Definition 2

2.3 Simplified Segment Definition 3

2.3.1 All Segments 3

2.4 Full Segment Definition 3

2.5 Program Execution 3

2.6 Build Your Program 4

2.7 Control Transfer Instruction 4

2.8 Conditional Jumps 4

2.9 Subroutines and Call Statement 4

2.10 Data Types and Definition 5

2.11 More about Variables 5

2.12 More about Labels 5

2.13 PTR directive 5

2.14 .COM executable 6

3 Arithmetic Operation **6**

3.1 Unsigned Addition 6

3.2 Unsigned Subtraction 6

3.3 Multiply and Division 6

3.4 Logic Instruction 7

3.5 BCD & ASCII Numbers Conversion 7

3.6 Rotate 7

3.7 Conditional Jump 7

1 Programming Language

- machine language: binary code
- assembly language: mnemonics for machine code instructions
- high-level language

2 Form of a Statement

- [label:] mnemonic operands [; comment]

- label: letters, 0-9, ? . @ \

2.1 Shell of a Real Program

- full segment definition
- simplified segment definition

2.2 Model Definition

- small: code, data $\leq$ 64KB
- medium: data $\leq$ 64KB, code > 64KB
- compact: code $\leq$ 64KB, data > 64KB

- large, huge, tiny

2.3 Simplified Segment Definition

- `.code`, `.data`, `.stack`
- correspond to CS, DS, SS
- DOS determines CS and SS registers automatically
- DS has to be manually specified

2.3.1 All Segments

- refer to slides
- procedure definition
  - label PROC [FAR|NEAR]
  - label ENDP
- END MAIN (MAIN is the entry)
- set DS
  - MOV AX, @DATA
  - MOV DS, AX

2.4 Full Segment Definition

- label SEGMENT
- label ENDS
- end program
  - mov ah, 4ch
  - int 21h
- you can compare this to syscall in OS

2.5 Program Execution

- call
- ret

2.6 Build Your Program

- editor program (.asm)
- assembler program (.lst, .crf, .obj)
- linker program(.obj, other obj -> .exe, .map)
- commands
  - MASM A:MYFILE.ASM
  - LINK A:MYFILE.OBJ

2.7 Control Transfer Instruction

- range
  - short (-128~127)
  - near (-32768~32767), control transferred in same code segment
  - far
    - CS/IP all changed
    - control transferred outside current code segment
- jumps
- call statement

2.8 Conditional Jumps

- don't memorize it. guess what flag registers are with the name.

2.9 Subroutines and Call Statement

- range
  - near
  - far
- wrap proc with proc / endp
- call is used to call a subroutine
  - ret is put at the end
  - push next instruction PC to stack
  - and then jump
- ret
  - pop PC from stack

- calling far proc
    - first CS, then IP pushed to stack
    - ret is auto expanded to retf

2.10 Data Types and Definition

- ORG 10
- x DB 12 (byte-size chunks)
- y DB 23H, 48H
- z DB 'Good Morning!'
- DB grows from low address to high
- DW (16-bit), DD, DQ
- NUM EQU 234 (num constant)
- x DB 6 DUP 23H (duplicate characters)
- y DW 3 DUP(0FF10H) (why not FF10H? distinguish from identifier)

2.11 More about Variables

- variable names have 3 attributes
    - segment value
    - offset address (+seg = logical address)
    - type
- get the segment value `SEG var`
- get the offset `OFFSET time`, or `LEA AX, time`

2.12 More about Labels

- implicit: `AGAIN: ADD AX, 03423H`
- explicit: `AGAIN LABEL FAR`
- attributes: segment, offset, type

2.13 PTR directive

- DATA1 DB 10H, 20H, 30H
- MOV BX, WORD PTR DATA (BX=2010H)
- MOV WORD PTR [BX], 10H ([BX:BX+1] <- 0010H)

- JMP FAR PTR label

2.14 .COM executable

- data and code segment together
- less than 64KB

3 Arithmetic Operation

3.1 Unsigned Addition

- `ADD dest, src`
- dest: reg, mem
- src: reg, mem, immediate
- no mem-mem
- change ZF, SF, AF, CF, OF, PF

- `ADC dest, src` dest += src + CF

3.2 Unsigned Subtraction

- `SUB dest, src` dest = dest - src
- `SBB dest, src` dest = dest - src - CF
  - takes 2's complement
  - add to dest
  - invert the carry

3.3 Multiply and Division

- `MUL operand`
  - byte: AL, op $\rightarrow$ AX
  - word: AX, op $\rightarrow$ DX:AX
  - word $\times$ byte: AH=0, AL, op $\rightarrow$ DX:AX
- `DIV denominator`
  - denominator cannot be zero
  - quotient cannot be too large

    - byte / byte: AL/op=AL. . . AH
    - word / word: AX / op = AX. . . DX
    - word / byte: AX / op = AL. . . AH
    - double-word / word: DX:AX / op = AX. . . DX
    - op can be register or memory

3.4 Logic Instruction

- AND dest, src
- SHR dest, times
    - 0 - MSB - LSB - CF
    - times = 1: SHR xx, 1
    - otherwise, SHR xx, CL
- SHL dest, times
    - CF - MSB - LSB - 0

3.5 BCD & ASCII Numbers Conversion

- refer to slides

3.6 Rotate

- ROR dest, times
- ROL dest, times
    - CF - MSB - LSB $\leftarrow$ CF

3.7 Conditional Jump

- refer to slides