Assembly Language

- Review of 80x86 architecture
- Overview Of 80x86 Assembler
- Procedures and the Stack
- Addressing Modes
An Assembly Program

COMMENT | Bubble sort procedure BBLSORT.

Objective: To implement the bubble sort algorithm

Inputs: A pointer to the array to be sorted
and its size are received via the stack.

Output: Returns nothing but the array is sorted
in ascending order.

SORTED EQU 0
UNSORTED EQU 1

.MODEL SMALL
.CODE
.486
PUBLIC _bubble_sort
_bubble_sort PROC
; save registers used by the procedure
pusha
mov BP,SP

; CX keeps the number of comparisons
; to be done in each pass. Note that CX is decremented
; by 1 after each pass.
mov CX, [BP+20] ; load array size into CX
mov BX, [BP+18] ; load array address into BX
next_pass:
    dec    CX         ; if # of comparisons is zero
    jz     done       ; then we are done
    mov    DI,CX      ; else start another pass

;DX is used to keep SORTED/UNSORTED status
mov    DX,SORTED   ; set status to SORTED

;SI points to element X and SI+2 to the next element
mov    SI,BX       ; load array address into SI
pass:
; This loop represents one pass of the algorithm.
; Each iteration compares elements at [SI] and [SI+2]
; and swaps them if ([SI]) < ([SI+2]).
mov AX,[SI]
cmp AX,[SI+2]
jg swap
increment:
; Increment SI by 2 to point to the next element
add SI,2
dec DI
jnz pass

cmp DX,SORTED ; if status remains SORTED
je done ; then sorting is done
jmp SHORT next_pass ; else initiate another pass
swap:
; swap elements at [SI] and [SI+2]
            xchg    AX,[SI+2]
            mov     [SI],AX
            mov     DX,UNSORTED    ; set status to UNSORTED
            jmp     SHORT increment

done:
; restore registers
            popa
            ret

_bubble_sort ENDP
END
The Pentium Registers

• Data Registers,

• Pointer and Index registers,

• Control Registers

• Segment Registers
Memory

- An array of bytes,

- Read cycle,

- Write cycle,

- Storing multibyte data.
Pentium Memory Architecture

• Segment

• Offset

• Real Mode

• Protected Mode

• Segment Registers
I/O

- I/O Ports
- I/O controllers
- I/O devices
Assembly Language Statements

- **Format**

  
  [label]  mnemonic  [operands]  [;comment]

- **Data Allocation**

  [variable-name]  define-directive  initial-value
Where are the operands?

- In a register
- in the instruction
- in main memory
- at an I/O port
Addressing Modes

- Register Addressing: mov EAX, EBX

- Immediate Addressing: mov AL, 75

- Direct addressing: mov AL, response

- Indirect addressing:

  mov BX,OFFSET table1
  mov [BX], 100
  add BX,2
  mov [BX], 99
Data transfer Instructions

- **mov** destination, source

  There are restrictions on the destination source pairings. In particular both cannot be in memory.

- **xchg** loc1,loc2

  Here again both locations cannot be in memory.
Arithmetic Instructions

- inc   destination
  dec   destination

  destination can be any 8-, 16- or 32-bit operand.

- add destination, source

  Both operands cannot be in memory

- sub destination, source

- cmp loc1,loc2

  Subtracts the second operand from the first, but the result is discarded, only the flags are set.
Conditional Execution

- jmp label

- Other jumps

  je jump if equal

  jg jump if greater

  jl jump if less

  jge jump if greater or equal

  jle jump is less or equal

  jne jump if not equal
Jumps based on flag values

jz jump if ZF=1

jnz jump if ZF=0

jc jump if carry (CF=1)

jnc jump if CF=0 r
Iteration I

mov CL,50 ; will loop 50 times
repeat:
   (loop body)
   dec CL
   jnz repeat
Iteration 2 (Pentium only)

    mov     CX,50

repeat:

    ....

    (loop body)

    ...

    loop repeat
Logical Instructions

• and destination, source

• or destination, source

• not destination
Shift Instructions

- shl destination, count
  shl destination, CL

- rol destination, count
  rol destination, CL

- rcl destination, count
  rcl destination, CL

There is no carry. CF catches the last bit

The carry flag is included in the process.
Defining Constants

name equ expression

Symbols defined by EQU cannot be redefined. Whereas one can use = and these can be re-defined.

name = expression