# EEE 332/ CSE 331 Lab 3

Topics to be covered in class today:

* Conditional Jumps/Unconditional Jumps
* Procedures
* Instructions: CMP, AND, SUB, JZ, JMP

|  |  |  |
| --- | --- | --- |
| Instruction | Operands | Description |
| CMP | REG, memory  memory, REG  REG, REG  memory, immediate  REG, immediate | Compare.  Algorithm:  operand1 - operand2  Result is not stored anywhere, flags are set (OF, SF, ZF, AF, PF, CF) according to result.  Example:  MOV AL, 5  MOV BL, 5  CMP AL, BL ; AL = 5, ZF = 1 (so equal!)  RET |
| AND | REG, memory  memory, REG  REG, REG  memory, immediate  REG, immediate | Logical AND between all bits of two operands. Result is stored in operand1.  These rules apply:  1 AND 1 = 1  1 AND 0 = 0  0 AND 1 = 0  0 AND 0 = 0  Example:  MOV AL, 'a' ; AL = 01100001b  AND AL, 11011111b ; AL = 01000001b ('A')  RET |
| SUB | REG, memory  memory, REG  REG, REG  memory, immediate  REG, immediate | Subtract.  Algorithm:  operand1 = operand1 - operand2  Example:  MOV AL, 5  SUB AL, 1 ; AL = 4  RET |
| JZ | Label | Short Jump if Zero (equal). Set by CMP, SUB, ADD, TEST, AND, OR, XOR instructions.  Algorithm:  if ZF = 1 then jump  Example:  include 'emu8086.inc'  ORG 100h  MOV AL, 5  CMP AL, 5  JZ label1  PRINT 'AL is not equal to 5.'  JMP exit  label1:  PRINT 'AL is equal to 5.'  exit:  RET |
| JMP | Label | Unconditional Jump. Transfers control to another part of the program. 4-byte address may be entered in this form: 1234h:5678h, first value is a segment second value is an offset.  Algorithm:  always jump  Example:  include 'emu8086.inc'  ORG 100h  MOV AL, 5  JMP label1 ; jump over 2 lines!  PRINT 'Not Jumped!'  MOV AL, 0  label1:  PRINT 'Got Here!'  RET |

## Procedures

Procedure is a part of code that can be called from your program in order to make some specific task. Procedures make program more structural and easier to understand. Generally procedure returns to the same point from where it was called.

The syntax for procedure declaration:

name PROC

; here goes the code

; of the procedure ...

RET

name ENDP

name - is the procedure name, the same name should be in the top and the bottom, this is used to check correct closing of procedures.

Probably, you already know that RET instruction is used to return to operating system. The same instruction is used to return from procedure (actually operating system sees your program as a special procedure).

PROC and ENDP are compiler directives, so they are not assembled into any real machine code. Compiler just remembers the address of procedure.

CALL instruction is used to call a procedure.

Example:

ORG 100h

MOV AL, 1

MOV BL, 2

CALL m2

CALL m2

CALL m2

CALL m2

RET ; return to operating system.

m2 PROC

MUL BL ; AX = AL \* BL.

RET ; return to caller.

m2 ENDP

END

## Task 1

Copy, compile and run the following code:

org 100h

jmp adder

printer:

mov ah,2

mov dl,al

add dl,'0'

int 21h

jmp finish

adder:

mov al,2

mov bl,2

add al,bl

jmp printer

finish:

ret

## Task 2

Copy, compile and run the following code:

org 100h

lea si,arr

mov cx,5

search\_loop:

mov al,[si]

cmp al,key

JZ found

inc si

LOOP search\_loop

mov dx,offset error

mov ah,9

int 21h

ret

found:

lea di,arr

mov ax,si

mov bx,di

sub ax,bx

add ax,'0'

mov dx,ax

mov ah,2

int 21h

ret

arr db 1,2,3,4,5

key db 2

error db "Key not found$"

## Task 3

Write a program that will count the number of characters in a string.

## Task 4

Write a program that will concatenate (join) two strings. Make sure the input strings are not destroyed and the final answer must be inside a third array. Input from user not required. Create two strings in your program.

Example:

String 1: “Hello World, “

String 2: “this is Assembly Language Programming”