CS1520 Tutorial 2

AX is made of AH and AL. It is used for input/output and arithmetic instructions such as multiplication.

BX is made of BH and BL, it is the base register used in indexed addressing

CX is the count register, it stores the loop count for iterative operations.

DX is the data register, used for input and output. Used with the AX register to multiply and divide with larger values.

Set bits- make them 1

Clear bits- make them 0

Complement bits- Change value from 0 to 1 or 0 to 1

Manipulating bits makes programs more effective. Smaller, less Ram, faster, storage.

Logical instructions: and, or, xor, not, test

The mul instruction takes a single parameter, the other operand must be ax or al. Results are stored in ax if 8-bit numbers are multiplied. If 16-bit numbers are multiplied, the result is split between dx and ax respectively.

Mov al,7

Mov bl,7

Mul bl

Ax = al\*bl = 63

The –div. You can divide 8-bit number into a 16-bit number

Number divided is stored into ax

Divide 16-bit number into a 32-bit number

32-bit number is stored into ax and bx registers.

To clear bit 5 of a byte, we use the ‘and’ method with 1101 1111. Because the 0 is changed with the and function. To clear a bit or group of bits, you define a mask consisting of 0s for the bits to be cleared and 1s for everywhere else. This is called masking.

To set bits to 1 regardless of their current setting, we use the ‘or’ method. E.g. mask bits 4 and 5 we use 0011 0000

Mov al, 7 ; 7 is 0000 0111

Or al, 0011 000b

; 7 becomes 0011 0111

Or can be used to convert the capitalization of a letter. – The 6th bit.

To toggle value of bits we use the ‘xor’ method. The bit mask to toggle has 1s for bits to toggle and 0s to keep the same.

Eg. Mov al, 67h ; al = 0011 0111

Xor al, 08h ; xor it with 0100 0011

; al is 34h, 0111 0100

Xor can be used to clear registers.

Xor cx, cx

Mov can clear a register

Mov cx, 0

Sub can clear register

Sub cx,cx

The not instruction inverts all of its bits and its operand

Mov al, 33h

Not al

Al = 0011 0011

Not al= 1100 1100

To display something, you use the sub-program 2h and the value is stored into the ah register.

Mov dl, ‘a’

Mov ah, 2h

Int 21h ; display character

Inc method increments value in register.

e.g.

ax = 2

if (( ax – bx ) != 0) {

ax = ax +1;

}

bx = bx +1

in Assembly language:

mov ax,2

sub ax,bx

jz next1

inc ax

next1:

inc bx

The cmp method is used to perform comparison. It’s identical to the sub instruction except it does not affect operands.

To test if a particular bit is set to 1 or 0, we can use the and instruction and mask which clears the remaining bits.

E.g. if we test if bit 1 of al is 1 or 0, we use a mask with bit 1 set, the mask: 0000 0010

And al, 0000 0010b

Jz bit1\_clear ;go to bit1\_clear if bit1 == 0

Bits can be moved right or left.

Values shifted off the en of an operand are lost.

Shift moves bits a specified number or places right or left

Rotate moves bits a specified number of places to the right or left. For each bit rotated, the last bit in the direction of the rotate is moved into the first bit position at the end of the operand.

Shifts and rotations are used for multiplication and division.

E.g. Multiply by 100, shift the number left one digit and add a 0 on the right side.

Shift left one bit is equivalent to multiplying by 2. Shifting 2 bits left multiplies by 4.

A right shift of n bits divides a number by 2n,

The ***sar*** (shift arithmetic right) instruction if we want to preserve the sign bit.

Dividing al by 2: (al=12)

Mov al,12

Sar al,1

Sh1 ( Shift left)

Shr (Shift bits right)

Sar (signbit shifts into itself)

# Rotate

Ror = rotate right

Rol = rotate left.

If we rotate the al register by 2 bits, then bit 7 rotates to position 1, bit 6 rotates to position 0

If al contains 0110 0011,

Rotating ror,1 makes al contain

0110 011 => 1011 0001

Registers are temporary storage in the CPU that holds the data the processor is currently working on, while memory holds the program instructions and the data the program requires.

Data has to be loaded on a CPU register from memory before the CPU can process it. Memory is much slower than registers, the fastest operations work directly with registers.

There is a lot more memory than registers, there are generally only a few registers available n a processor.

E.g. Stack register, flags register, program counter and some addressing registers.

# Tutorial 3:

Questions

1)

mov al, 62h

and al, 1101 1111b

2)

mov al,’b’

and al, 0dfh

0dfh = 1101 1111

3)

mov ax,0

and ax, 0000 0000b

xor ax,ax

sub ax,ax

xor and sub don’t use any other registers, they use 2 bytes so they are the fastest.

4)

5)

al, 63h =0110 0011 = 99

ror al,1 = 1011 0001 = 177

6)

mov al,12

sar al,2

7)

complement register al which contains 33h

8)

sal ax,4

sal bx,6

9)

sar ax,2

sar dx,5