

# ASSEMBLY LANGUAGE

Abhijite Deb Barman(CSE'20 HSTU)

## Template

```
.model small      ;can be large,tiny,etc but small is enough for academic
.stack 256        ;memory size
.data             ; data segment. We can declare variable
msg db 'Bangladesh $' ;declared variable as name 'msg' db=define byte
msg1 db 'num1 $'
msg2 db 'num2 $'
num1 db 5         ;declare constant variable num1
num2 db ?         ;declare unknown variable num2
.code             ;code segment start
```

## main proc

```
    mov Ax,@data  ;data loaded to accumulator
    mov ds,Ax     ;accumulator to data segment
```

```
    exit:         ;next 5 lines needed to end code
    mov ah,4ch
    int 21h
    main endp
```

## end main

## Input

```
mov ah,1          ;for single key input always need this and stored in "al" reg
int 21h
mov bl,al         ;move "al" to "bl" as input loads in "al"
```

## Output(character)

```
mov ah,2          ;for number or key output always need this
mov dl,bl         ;print output only from data segment so move before print
int 21h
```

**Output(number)**

```
mov ah,2  
mov dl,num1  
add dl,48 ;it always print ascii values so 5+48=53(5)  
int 21h
```

**Print message**

```
mov ah,9 ;for string output always need this  
lea dx,msg ;lea=load effective area and string msg copied to whole dx(dh+dl)  
int 21h
```

**Newline**

```
mov ah,2  
mov dl,0Ah ;0Ah=10=newline we can use mov dl,10 also  
int 21h  
mov dl,0Dh ;0Dh=13=carriage return we can use mov dl,13 also  
int 21h
```

**Exchange**

```
Xchg bl,bh
```

**Add two number**

```
add bh,bl  
sub bh,48
```

**Subtract two number**

```
Sub bh,bl  
Add bh,48
```

**Lower to upper**

```
sub bl,32
```

**Upper to Lower**

```
Add bl,32
```

**Multiplication of two numbers**

```
mov al,3  
mov bl,2  
mul bl ;al=al*bl
```

### For big number

```
mul a
aam          ;adjust ax after multiplication
mov bx,ax
```

```
mov ah,2
mov dl,bh    ;print from upper bit segment
add dl,48
int 21h
```

```
mov dl,bl    ;print from lower bit segment
add dl,48
int 21h
```

### Division

```
mov al,14
mov bl,3
```

```
div bl ;al=al/bl    ;result stores in al and remainder stores in ah
mov cl,al
mov ch,ah
```

### Conditional(only if)

```
Jg-->jump greater than
Jge-->jump greater than or equal
Jl-->jump less than
Jle-->jump less than or equal
Jng-->jump not greater than
Jnge-->jump not greater than or equal
Jnl-->jump not less than
Jnle-->jump not less than or equal
jz-->jump zero
Jnz-->jump not zero
```

```
cmp bl,bh
jle L1
jmp L2
```

```
L1:
  mov ah,2
  mov dl,bl
  add dl,48
  int 21h
```

```
L2:
If else
cmp al,ah
jge L1
jmp L2
```

```
L1:
  mov ah,9
  lea dx,a
  int 21h
  jmp exit:
```

```
L2:
  mov ah,9
  lea dx,b
  int 21h
  jmp exit:
```

### **If, else-if, else**

```
cmp bl,53(compare with 5(ascii 48+5=53))
jg L1
jl L2
jmp L3
```

```
L1:
  mov ah,9
  lea dx,a
  int 21h
  jmp exit:
```

```
L2:
mov ah,9
lea dx,b
int 21h
jmp exit:
```

```
L3:
mov ah,9
lea dx,c
int 21h
jmp exit:
```

### **OR Operation**

```
cmp bl,'Y'
jz L1
```

```
cmp bl,'y'
jz L1
```

```
jnz loop exit:
```

```
L1:
mov ah,2
mov dl,bl
int 21h
```

### **Loop(increment)**

```
mov cl,'1'
```

```
for:
cmp cl,bl
jg exit:    ;if cl>bl exit
```

```
mov ah,2 ;statement
mov dl,cl
int 21h
```

```
inc cl      ;increment cl value
jmp for:    ;call for again
```

### **Loop(Decrement)**

```
mov cl,bl
```

```
for:
```

```
cmp cl,'0'
```

```
je exit:
```

```
mov ah,2
```

```
mov dl,cl
```

```
int 21h
```

```
dec cl
```

```
jmp for:
```

### **Another way for “for loop”**

```
mov ah,1
```

```
int 21h
```

```
mov bl,al
```

```
sub al,48
```

```
mov cx,0 ;counter=0
```

```
mov cl,al ;al=8 bit cl=8bit keep al in cl
```

```
for:
```

```
mov ah,2
```

```
mov dl,bl
```

```
int 21h
```

```
dec bl
```

```
loop for:
```

**Print A-Z**

```
mov bl,'a'
```

```
mov cx,0  
mov cl,26
```

```
for:
```

```
mov ah,2  
mov dl,bl  
int 21h
```

```
inc bl
```

```
loop for:
```

**While Loop**

```
mov ah,1  
int 21h  
mov bl,al
```

```
mov cl,'1'
```

```
while:
```

```
cmp cl,bl  
jg exit:
```

```
mov ah,2  
mov dl,cl  
int 21h
```

```
inc cl;
```

```
jmp while:
```