CMSC 131 website:

https://sites.google.com/site/cmsc131sem1yr1213/

III. STRUCTURED ASSEMBLY LANGUAGE PROGRAMMING TECHNIQUES

Control Transfer Instructions

Objectives

At the end of the discussion, the students should be able to:

- Implement selection statements in assembly, and
- Describe how unconditional jumps and conditional statements work

Control Transfer Instructions

- allows program control to transfer to specified label
- Unconditional or Conditional
- Unconditional
  - executed without regards to any situation or condition in the program
  - transfer of control goes from one code to another by force

    jmp label – unconditional jump

Control Transfer Instructions

```
mov al, 5
add [num1],al
jmp next


mov eax, 4
mov ebx, 1
mov ecx, num1
mov edx, 1
int 80h
```

(1)

```
next:
mov eax, 4
mov ebx, 1
mov ecx, num2
mov edx, 1
int 80h
```

(2)

Control Transfer Instructions

- Conditional
  - a jump carried out on the basis of a truth value
  - the information on which such decisions are based is contained in the flags registers

Boolean Expressions

- evaluates to True or False
- compares two values

- **cmp** source1, source2
- Source1 may be a register or memory
- Source2 may be a register, memory or immediate
- Operands cannot be both memory.
- Operands must be of the same size.

Conditional Jumps

- usually placed after a **cmp** instruction
  conditional_jump label

- JE – branches if source1 == source2
- JNE – branches if source1 ≠ source2

Conditional Jumps

- Signed Conditional Jump
  - JL or JNGE
    - branches if source1 < source2
  - JLE or JNG
    - branches if source1 ≤ source2
  - JG or JNLE
    - branches if source1 > source2
  - JGE or JNL
    - branches if source1 ≥ source2

Conditional Jumps

- Unsigned Conditional Jumps
  - JB or JNAE
    - branches if source1 < source2
  - JBE or JNA
    - branches if source1 ≤ source2
  - JA or JNBE
    - branches if source1 > source2
  - JAE or JNB
    - branches if source1 ≥ source2

Signed or Unsigned

```
mov al, FFh
cmp al, 10
jb label
```

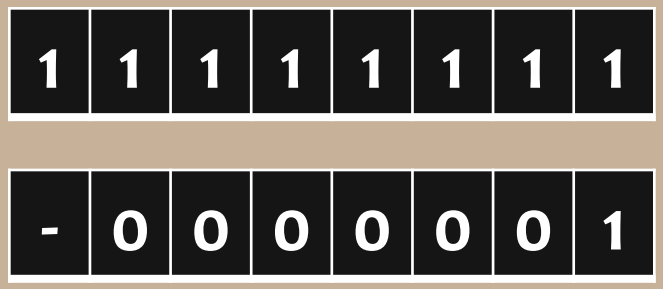
- FFh == 255
- will not jump to label

Signed or Unsigned

mov al, FFh

cmp al, 10

jl label



- FFh == -1
- will jump to label

Control Structure: IF Statement

```
if (boolean)
    {statements;}

if(AX>CX){
    BX = DX + 2;
}
```

```
cmp AX, CX
jg if_statement
jmp next_statement

if_statement:
    add DX, 2
    mov BX, DX
next_statement:
    ...
```

Better Design: Save on Jumps

```
cmp AX, CX
jg if_statement
jmp next_statement

if_statement:
    add DX, 2
    mov BX, DX
next_statement:
    ...
```

```
cmp AX, CX
jng next_statement

if_statement:
    add DX, 2
    mov BX, DX
next_statement:
    ...
```

Control Structure: IF-ELSE Statement

```
if (boolean)
{statements;}
else
{statements;}
```

```
if(AX>CX){
    BX = DX + 2;
} else {
    BX = DX – 2;
}
```

```
cmp AX, CX
jg if_statement
jmp else_statement
if_statement:
    add DX, 2
    mov BX, DX
    jmp next_statement
else_statement:
    sub DX, 2
    mov BX, DX
next_statement:
    ...
```

Better Design: Save on Jumps

```
cmp AX, CX
jg if_statement
jmp else_statement
if_statement:
    add DX, 2
    mov BX, DX
    jmp next_statement
else_statement:
    sub DX, 2
    mov BX, DX
next_statement:
    ...
```

```
cmp AX, CX
jng else_statement

if_statement:
    add DX, 2
    mov BX, DX
    jmp next_statement
else_statement:
    sub DX, 2
    mov BX, DX
next_statement:
    ...
```

Compound Boolean Expressions

- ANDed expressions
  - P and Q
  - True if and only if both expressions are True
- ORed expressions
  - P or Q
  - False if and only if both expressions are False

Short Circuited Evaluation

- ANDed Expressions
  - if the first expression is False, there is no need to check the second expression.
- ORed Expressions
  - If the first expression if True, there is no need to check the second expression.

Short Circuited Evaluation

- P and Q

  if (P == FALSE) then proceed to ELSE-part

  else

  if (Q == FALSE) then proceed to ELSE-part

  else

  proceed to THEN-part

Short Circuited Evaluation

- P or Q

  if (P == TRUE) then proceed to THEN-part

  else

  if (Q == TRUE) then proceed to THEN-part

  else

  proceed to ELSE-part

ANDed Expressions

```
if(AX >= 100) &&
(AX <=120) {
    BX = AX;
} else {
    BX = CX;
}
```

```
cmp AX, 100
jge other_cond
jmp else_stmt
other_cond:
    cmp AX, 120
    jle if_stmt
    jmp else_stmt
if_stmt:
    mov BX, AX
    jmp next_stmt
else_stmt:
    mov BX, CX
next_stmt:
    ...
```

ANDed Expressions

```
if(AX >= 100) &&
(AX <=120) {
   BX = AX;
} else {
   BX = CX;
}
```

```
cmp AX, 100
jnge else_part
cmp AX, 120
jnle else_part
then_part:
    mov BX, AX
     jmp next_part
else_part:
    mov BX, CX
next_part:
    ...
```

ORed Expressions

```
if(AX < 100) ||
(AX > 120) {
    BX = CX;
} else {
    BX = AX;
}
```

```
cmp AX, 100
jl then_part
cmp AX, 120
jng else_part
then_part:
    mov BX, CX
    jmp next_part
else_part:
    mov BX, AX
next_part:
    ...
```

Practice Exercise

```
if ((a1>=b1) && (a1<10)){
    printf("%s",msg1);
}
else {
    printf("%s",msg2);
}
c1=b1;
```