American International University-Bangladesh (AIUB)
Spring 2019-2020
Computer Architecture and Organization
Final Term: Assignment 1(Replacement of QUIZ) Assignment weight 20%

Mode of submission: Hand Written scan soft copy
Deadline of Submission: 25/3/20
Late submission will not be accepted

| | | | |
|---|---|---|---|
| Name: JOY KARMOKAR | ID: 18-39263-3 | Section: I | Submit:23/03/2020 |

1. Write the summary of chapter 6 (6.1 to 6.4)
2. Explain two types of JUMP.
3. Explain range of conditional jump. How CPU implements conditional jump. Descries tree types of conditional jump with a Table format?
4. Differentiate signed vs unsigned jump. Explain signed vs unsigned jump while working with characters.
5. Define branching. Describe branching structures with example for, **IF-THEN, IFTHEN-ELSE, CASE**.
6. Define Looping. Describe branching structures with example for **For LOOP, While LOOP and Repeat LOOP.**
7. What is the difference between while and repeat loop. What is the name of repeat loop in high level language.
8. Exercise 1 (a, c, e), 2, 3, 4(c), 5, 6,7, 8.

## Ans to the q.no -1

For assembly language to carryout useful tasks, there must be a way to make decesions and repeat sections of code. The Jump and loop instructions transfer control to another part of the program. This tras can be unconditional or can depend on a particular combination of status flag setting. There is different types of Jumps for single flag including JE or JZ, JNZ/JNE, JC, JNC, JO, JNo, JS, JNS, JP, JNP and loops are for, while and Repeat. The case structure, branching is controlled by an expession, The branches correspond to the possible values of expression.

Ans to the Q.no-2

Two types of Jump: (i) condition Jump
(ii) Uncondition Jump

Condition Jump: Conditional Jump Instruction are and important aspect of the decision are making process in programming. The conditional Jump instruction check the flag conditions are make decisions to change or not to change the sequence of the program.

Uncondition Jump: Uncondition Jump Instruction is executed, the Jump always ready to take pace to change the execution sequence. This is perforemed by the Jump instruction. Conditional exicution often involves a transfer of control to the address of an instruction that does not follow the currently executing instruction.

## Ans to the Q.no-3

Range of conditional Jump: The structure of the machine code of a conditional Jump instruction requires that destination must proceed the Jump instruction by not more than 126 bytes.

How CPU implements conditional Jump:

1. CPU looks at flag register.
2. Flag reflects the last thing processor did.
3. If codition for Jump is true CPU adjust the IP to the point destination level.
4. If condition for Jump is false IP is not altered, this mean instruction in the line will be done.

types of condition Jump:

signed Conditional Jumps

| | | |
|---|---|---|
| JG/JNLE | - Jump if greater than<br>- Jump if not less than or equal | ZF=0 and SF=OF |
| JGE/JNL | - Jump if greater than or equal<br>- Jump if less than or equal | SF=OF |
| JL/JNGE | - Jump if less than<br>- Jump if greater than or equal | SF<>OF |
| JLE/JNG | - Jump if less than or equal<br>- Jump if not greater than | ZF=1 or SF<>OF |

Unsigned conditional Jump

| | | |
|---|---|---|
| JA/ JNBE | - Jump if above<br>- Jump if not below or equal | ZF=0 and CF=0 |
| JAE/ JNB | - Jump if above or equal<br>- Jump if not below | CF=0 |
| JB/ JNAE | - Jump if below<br>- Jump if not above or equal | CF=1 |
| JBE/ JNA | - Jump if below or equal<br>- Jump if not above | CF=1 or ZF=1 |

## Ans to the question no-4

Difference between signed and unsigned Jump:

| signed | Unsigned |
|---|---|
| 1. The variable that holds a positive to negative value from 0-127 | 1. Doesn't distinguish between positive and negative |
| 2. signed Jump operate ZF, SF and OF flags. | 2. operate on ZF and CF flags. |
| 3. Have range 128 to 127 | 3. Have range of 0-255 |

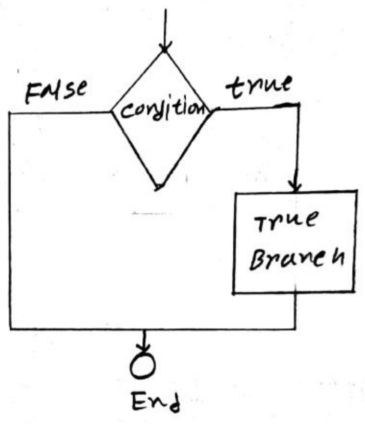
Each of the signed Jumps corresponds to an analogous unsigned Jump. For example the signed Jump JG and the Unsigned Jump JA. weather to use a signed or unsigned Jump depends on the interpretation given. In working with standard ASCII character set. Either signed or unsigned Jump may be used, because the sign bit of a byte containing a character code is always zero.

## Ans to the q.no-5

Branching structure enables a program to take path. depending or condition in high level languge.

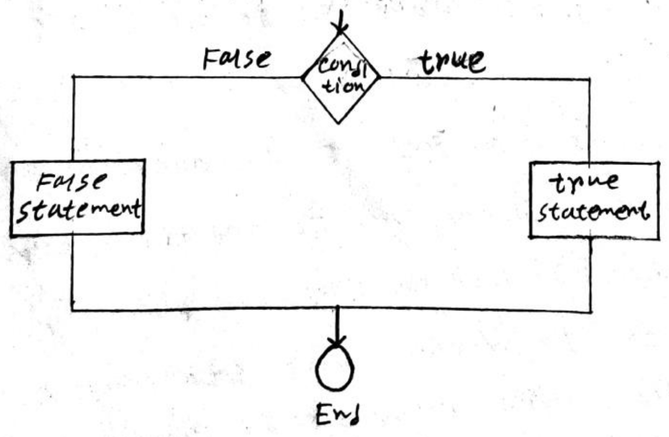
### If then:

If condition is true then execute true branch structure. If false nothing done

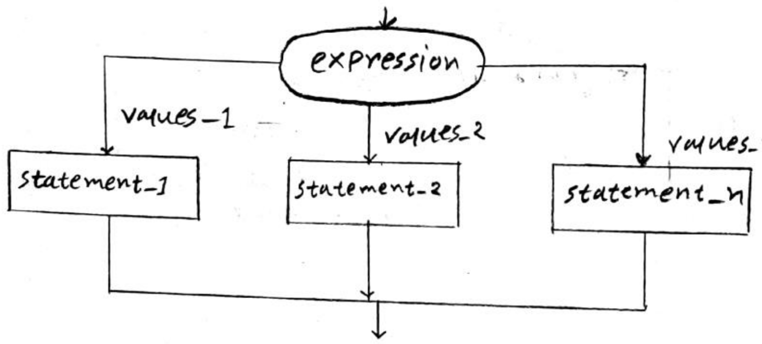


### If Then-Else:

If condition is true then execute true branch statement Else execute false branch statement.



case: If its value is a member of the set values-1, then statement-1 are executed. we assume that set value-1 . value-n are disjoint.

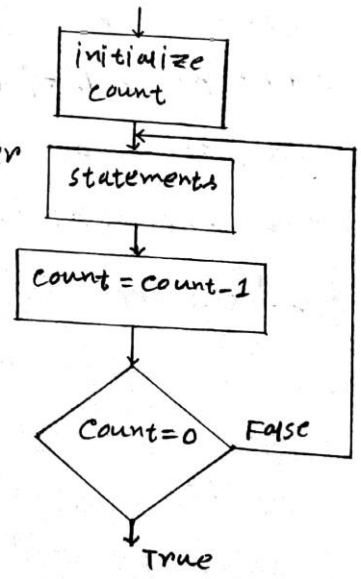


## Ans to the q.no-6

A loop is a sequence of instructions that is repeated

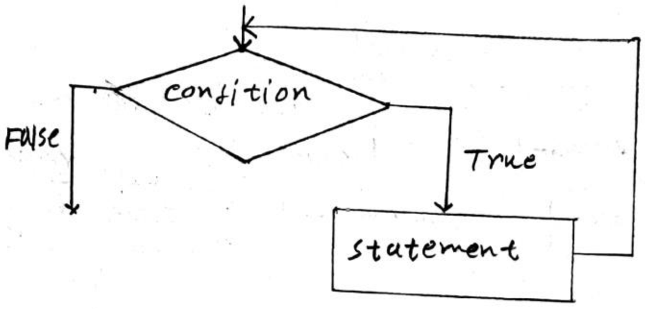
For loop:

The counter for the loop is the register CX which is initialized to loop-count. Execution of loop instruction cause CX to be decremeted automatically and if CX is not 0. control transfers to destination lebel.



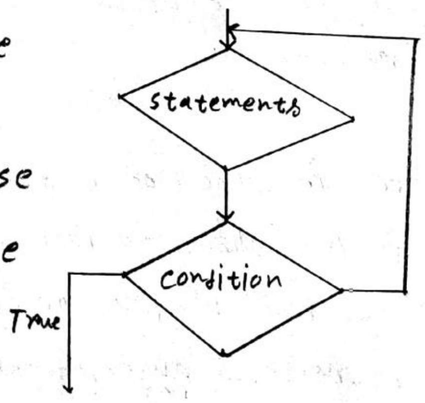
## While Loop :

The condition is checked at the top of the loop. If false the program goes on to whatever follows.



## Repeat :

Until loop the statements are executed and then the condition checked. If true the loop terminates. if false control branches to the top of the loop.



Ans to the Q.no-7

The difference between while and repeat loop.
Use of a while loop or a repeat loop is a matter of personal preference. The advantage of a while is that the loop can be bypassed if the terminating condition is initially false. Where as the statements in a repeat must be done at least once.
However, the code for a repeat loop is likely to be a little shorter because there is only a conditional Jump at the end. But a while loop has two jumps a conditional Jump at the top and a JMP at the bottom.

The name of repeat loop in high level language is Do loop.

Ans to the q.no-8

1/a.
```
cmp AX, 0
JGE End_if
MOV BX, -1
End_if: MOV AH, 4ch
        INT 21h
```

c)
```
cmp DL, 'A'
JL End-if
cmp DL, 'z'
JG End-if
MOV AH, 2
INT 21h
End-if: MOV AH, 4Ch
        INT 21h
```

e)
```
cmp AX, BX
JL Then_
Cmp BX, CX
JL Then_
mov Dx, 1
JMP END_IF
Then_: MOV BX, 0
END-IF: MOV AH, 4Ch
        INT 21h
```

3)
```
MOV AH, 2
INT 21H
Cmp AL, 'A'
JE EXE_CR
cmp AL, 'B'
JE EXE_LF
mov AH, 4Ch
INT 21H
EXE_CR; MOV AH, 2
        MOV DL, 0Dh
        INT 21H
EXE-LF: MOV AH, 2
        MOV DL, 0Ah
        INT 21H
```

3/a)
```
    MOV CX, 42
    MOV AX, 1
    MOV BX, 1
L1: ADD BX, 3
    ADD AX, BX
    LOOP L1
```

b)
```
    MOV CX, 19
    MOV AX, 100
    MOV BX, 100
L1: SUB BX, 5
    ADD AX, BX
    LOOP L1
```

4] c.

```
    MOV CX, 5
    MOV AH, 7
L1: INT 21H
    LOOP L1
    MOV DL, 'x'
    MOV CX, 5
    MOV AH, 2
L2: INT 21H
    LOOP L2
```

5]

```
MOV AX, 0
while_:
    CMP CX, BX
    JL END_while
    INC AX
    SUB CX, BX
    JMP while_
End_while: MOV AH, 4Ch
           INT 21h
```

6]

```
    XOR CX, AX
L1: ADD CX, AX
    DEC BX
    JNZ L1
```

7. a]

```
    MOV AH, 1
    MOV CX, 80
L1: INT 21H
    CMP AL, 20H
    LOOP L1
```

7. b]

```
    MOV AH, 1
    MOV CX, 80
L1: INT 21H
    CMP AL, 0DH
    LOOPNE L1
```

8)

```
MAIN PROC

    MOV AH,2
    MOV DL,'?'
    INT 21H
    MOV AH,1
    INT 21H
    MOV BL,AL
    INT 21H
    CMP BL,AL
    JG SWITCH
    JMP DISPIAY
SWITCH: XCHG AL,BL

DISPIAY: MOV AH,2
         MOV DL,0AH
         INT 21H
         MOV DL,BL
         INT 21H

  MOV DL,AL
  INT 21H
OUT: MOV AH,4Ch
     INT 21h

 MAIN ENDP
 END MAIN
```