Chapter 7

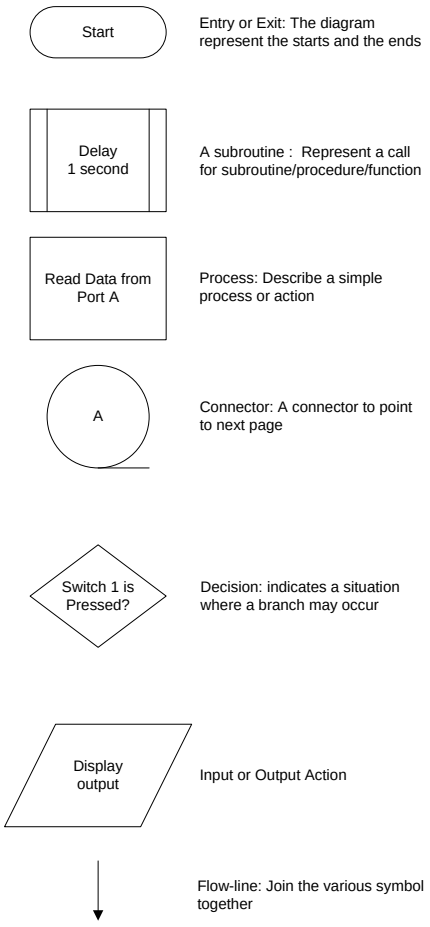
Program Flow

Expected Outcomes

- Interpret the flowchart into effective program
- Identify the type of flow control
- Differentiate between branch and jump instructions and know to use it
- Use various branch instructions to produce effective program
- Develop a program using branch and looping techniques

Introduction

- Normally, a program executes each instruction in sequential order, one after another
- Sometimes, the program uses a branch or jump instruction to redirect the program flow
- Thus, it changes the program counter to appropriate value
- This allows a program to skip over instructions or repeat a sequence of instructions



Program Flow

- There are two techniques
  - **Jump**
    Instruction : **`JMP`** (JUMP) and **`JSR`** (Jump to Subroutine)
  - **Branch**
    - Conditional
      - If condition (based on flag condition) is true, it redirects program flow
    - Unconditional
      - **`BRA`** (Branch Always) and **`BSR`** (Branch to Subroutine)

JMP Instruction

- When the instruction is executed, the current value of PC is replaced by the effective address (absolute address)
- **Syntax**

```
JMP   <label>
```

- Mode address:
  - **Absolute short**
  - **Absolute long**
  - **Indirect register address**
- Another form of jump is **JSR** where it is used to call a sequence of instruction
- **Syntax**

```
JSR   <label>
```

Unconditional Branch

- Unlike the Jump instruction, the effective address of the branch instruction is determined by adding the current PC with the offset
- The offset is a signed number and can be 8-bit (Short) or 16-bits (Long), thus allowing the branch to move forward or backward
- There are only two unconditional branch
  - **BSR**
  - **BRA**
- **Syntax**

```
BRA  <label>
```

```
BSR  <label>
```

Offset

- Basic formula for offset calculation

Offset = Destination Address – Current PC

```
2000                        ORG $2000
2000   6002     START       BRA  LABEL
2002   60FE                 BRA  *
2004   60FA     LABEL       BRA  START
2006                        END
```

Offset

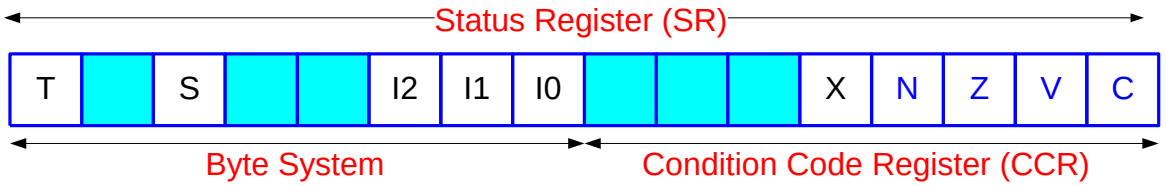
- Example

Address	Machine Code	Label	Instruction
0001 0000	606C	LOOP1	BRA.S LOOP2
0001 006E	6000 4002	LOOP2	BRA.L LOOP4
0001 4000	6000 BFFE	LOOP3	BRA.L LOOP1
0001 4072	608C	LOOP4	BRA.S LOOP3

Conditional Branch

- **Branch conditional code (Bcc)** is a conditional branch that based on the status of selected flag
- If the flag meets the requirement, branch will be executed; otherwise PC is unchanged and the next instruction is executed
- **Syntax**

```
Bcc.s      <label>
```



Conditional Branch

Instruction	Definition	Arithmetic	Condition
BEQ	Branch Equal to zero	U	Z=1
BNE	Branch Not Equal to Zero	U	Z=0
BMI	Branch Minus	U	N=1
BPL	Branch Plus	U	N=0
BCS	Branch Carry Set	U	C=1
BLO	Branch Lower	U	C=1
BCC	Branch Carry Clear	U	C=0
BHS	Branch Higher or Same	U	C=0
BVS	Branch Overflow Set	S	V=1

Conditional Branch

Instruction	Definition	Arithmetic	Condition
BGT	Branch Greater Than or Equal	S	Z.(N⊕V)=0
BLT	Branch Less Than or Equal	S	N⊕V=1
BGE	Branch Greater Than or Equal	S	N⊕V=0
BLE	Branch Less Than or Equal	S	Z.(N⊕V)=1
BHI	Branch Higher	U	C+Z=0
BLS	Branch Lower Than or Same	U	C+Z=1

Compare Instruction

- Its function is to compare between two data thus activate certain flags
- It is subtraction but the operand are not changed
- Various compare instructions
  - Compare
    `CMP.s <ea>,Dn`
  - Compare Immediate
    `CMPI.s #,<ea>`
  - Compare Address (size only W, L)
    `CMPA.s <ea>,An`
  - Compare Memory
    `CMPM.s (An)+,(An)+`

Example of CMP instruction

- **Example**: `CMP.W D3,D2`

D2	D3	D2-D3(W)	X N Z V C
0000 3040	0000 3040	0000	1 0 1 0 0
0000 3041	0000 3040	0001	1 0 0 0 0
0000 3040	0000 3041	FFFF	1 1 0 0 1
0000 0000	0000 FFFF	0001	1 0 0 0 1
0000 FFFF	0000 0000	FFFF	1 1 0 0 0
0000 9000	0000 7000	2000	1 0 0 1 0
0000 7000	0000 9000	E000	1 1 0 1 1

Conditional Branch

- Example : If the content of registers are as follow,

```
D0=$4A242000,D1=$0C00A180,D2=$7020E000,D3=$8A8C60F5
A1=$002C4B30,A2=$002C8000,CCR=%00000
```

Evaluate the following instruction set

Instruction stream		CCR	Bcc	Program Flow
MOVE.W	#$8C00,D0	01000	BHI LABEL	Yes
CMP.W	D2,D0	00001	BLS LABEL	Yes
ADD.B	D1,D3	10011	BLT LABEL	No
CMP.L	D0,D3	00010	BLE LABEL	Yes
CMPA	A1,A2	00000	BLO LABEL	No

More Examples..

- **Example 1:** Add D1 and D0. Branch to label EXIT if there is carry

```
            ADD.B D1,D0
            BCS EXIT
            :
EXIT
```

- **Example 2:** If D2 = D3, branch to SAME

```
            CMP.B D2,D3
             BEQ SAME
            :
SAME
```

More Examples..

- **Example 3**: Increment D0 if content of D1 is ‘A’ or ‘a’

```
        CMP.B    #'A',D1
        BEQ      SKIP
        CMP.B    #'A',D1
        BNE      SKIP2
SKIP    ADDQ.B   #1,D0
SKIP2
```

More Examples..

- **Example 4**: Subtract D1 from D2. If overflow or carry occurs, increment D3. Otherwise, decrement D3

```
        SUB.W   D1,D2
        BVS     INC
        BCS     INC
        SUBQ.B  #1,D3
        BRA     SKIP
INC     ADDQ.B  #1,D3
SKIP
```

More Examples..

- **Example 5**: Add absolute value of D2 to D3

```
        CMP.B   #0,D2
        BPL     LABEL
        NEG.B   D2
LABEL   ADD.B   D2,D3
```

More Examples..

- **Example 6**: Increment D7 if D0 contain a lowercase letter of character

```
        CMPI.B #'a',D0
        BLT    LABEL
        CMPI.B #'z',D0
        BGT    LABEL
        ADDQ.B #1,D7
LABEL
```

More Examples..

- **Example 7:** Convert content of D4 to a small letter (if it is a character)

```
            CMPI.B  #$41,D4
            BLT     LOWER
            CMPI.B  #$5A,D4
            BGT     LOWER
            ADD.B   #20,D4
LOWER
```

More Examples..

- **Example 8:** Clear D0 if the content of D5 is hexadecimal character

```
                CMPI.B  #'0',D5
                BLT     NOT_HEX
                CMPI.B  #'9',D5
                BLE     HEX
                CMPI.B  #'A',D5
                BLT     NOT_HEX
                CMPI.B  #'F',D5
                BGT     NOT_HEX
HEX             CLR.B   D0
NOT_HEX
```

More Examples..

- **Example 9:** To obtain smaller value of two numbers of M and N and store the result in MIN

```
MULA   MOVE.B      M,D0
       CMPB.B      N,D0
       BLE         OUT
       MOVE.B      N,MIN
       BRA         QUIT
OUT    MOVE        D0,MIN
QUIT   *
```

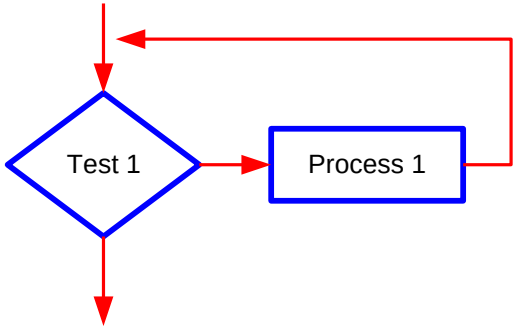
More Examples..

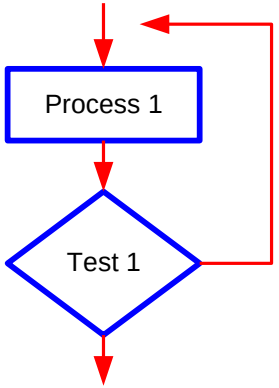
- **Example 10:** Squaring the difference between D0 and D1.

```
START   MOVE    D1,D2
        SUB.W   D0, D1
        BVS     UNSIGN
        BMI     SIGN
UNSIGN  MULU    D1,D1
        BRA     EXIT
SIGN    MULS    D1,D1
EXIT    BRA     *
```

Looping

- Certain segment of instruction needs to be executed more than one time
- Looping technique provides this mechanism to ensure the program is efficient
- There are several ways how the looping can be conducted
  - *Do-While* technique
  - *While* technique
  - *For* technique





More Examples..

- **Example 1:** Add the content of location $1000 - $10FF and place the answer in D0

```
START    MOVEA.L  #$1000,A0
         CLR.W    D0
         CLR.W    D1
AGAIN    MOVE.B   (A0)+,D1
         ADD.W    D1,D0
         CMPA.L   #$1100,A0
         BNE      AGAIN
```

More Examples..

- **Example 2:** Clear location $5000 - $5FFF

```
START   MOVEA.L #$5000,A1
REPEAT  CLR.L   (A1)+
        CMPA.L  #$6000,A1
        BNE     REPEAT
        BRA     *
```

More Examples..

- **Example 3:** Add the following numbers :1234, -1274, 9845, 122, -3432, 3333, 4,19

```
START     MOVEA.L #DATA,A0
          MOVE.B  #8,D0
          CLR.L   D2
AGAIN     MOVE.W  (A0)+,D1
          EXT.L   D1
          ADD.L   D1,D2
          SUBQ.B  #1,D0
          BNE     AGAIN

DATA      DC.W    1234,-1274,9845,122
          DC.W    -3432,3333,4,19
```

More Examples..

- **Example 4:** Perform the sum 1+2+3…+10 by using looping technique and store the answer in SUM

```
        ORG         $1000
SUM     DS.B        1
START   CLR         D2
        MOVE.B      #1,D1
AGAIN   ADD.B       D1,D2
        ADD.B       #1,D1
        CMP.B       #11,D1
        BNE         AGAIN
        MOVE.B      D2,SUM
```

More Examples..

- **Example 5:** Comparing to memory blocks and place $FF in D0 if it is equal. Otherwise, clear register D0

```
        ORG     $1000
        LEA     BLK1,A0   ;point to the beginning
        LEA     BLK2,A1   ;of block1 and 2
        MOVE.W  #SIZE,D0  ;initialize block size
LOOP    CMPM.L  (A0)+,(A1)+
        BNE     NE
        SUBQ.L  #1,D0
        BNE     LOOP
        MOVE.B  #$FF,D0
NE      CLR.B   D0
```

More Examples..

- **Example 6:** Reversing a string–swaps the bytes at the opposite end and moves the pointers towards the middle string until they meet

```
START   LEA     STRING1,A0
LOOP    TST.B   (A1)+
        BNE     LOOP
        SUBA    #1,A1
        LEA     STRING,A0
AGAIN   MOVE.B  -(A1),D0
        CMPA.L  A1,A0
        BHS     EXIT
        MOVE.B  (A0),(A1)
        MOVE.B  D0,(A0)+
        BRA     AGAIN
```

More Examples..

- **Example 7:** Write a program to scan a list of seven unsigned 16-bit integers. Choose the biggest number and place result in D0.

```
START   LEA       DATA,A0
        MOVEQ     #6,D1
        MOVE.W    (A0)+,D0
LOOP    CMP.W     (A0)+,D0
        BCC       SKIP
        MOVE.W    -2(A0),D0
SKIP    SUBQ      #1,D1
        BNE       LOOP
        BRA       *
DATA    DC.W      3454,100,2342,2342,3453
        DC.W      3444,2222
```

More Examples..

- **Example 8:** Write a program to scan a list of 5 signed 16-bit integers. Choose the biggest number and place result in D0

```
START LEA    DATA,A0
      MOVEQ #4,D1
      MOVE.W (A0)+,D0
LOOP  CMP.W (A0)+,D0
      BLT SKIP
      MOVE.W -2(A0),D0
SKIP  SUBQ #1,D1
      BNE LOOP
      BRA *

DATA  DC.W  -3454,2000,-2342,342,-4533
```

More Examples..

- **Example 9:** Write a program to add the absolute value of 5 signed words located in a list. The sum must be placed in D7

```
START       LEA      DATA,A0
            MOVEQ    #5,D0
            CLR.L    D7
LOOP        MOVE.W   (A0)+,D2
            BPL      SKIP
            NEG.W    D2
SKIP        EXT.L    D2
            ADD.L    D2,D7
            SUBQ     #1,D1
            BNE      LOOP
            BRA      *
DATA        DC.W    -4533,3444,-2222,575,123
```

Self-Test

Exercise

If D0=$0001000,D1=$0071A880,D2=$D01A9406,
D3=$2140FFFC,A0=$00002000,A1=$00004000,
A2=$00003FFC,A4=$00003004,CCR=%00000,
answer the following question.

Instruction stream	CCR	Bcc	Program Flow
SUB.W #1,D0		BHI LABEL	
CMP.W D1,D2		BGE LABEL	
ADD.B D1,D3		BVS LABEL	
SUB.W #$35CD,D3		BLE LABEL	
CMPA A1,A2		BLO LABEL	

Self-Test

Exercise

Translate the following code into the 68000 assembly language

```
IF A=8 THEN B=20
       ELSE C=A-6
```

Exercise

Write a program to convert the string of character to upper case. The string begins at address $3000 and the NULL character is used to signify the end of the string. Calculate the byte requirement and clock cycle for the program.

Self-Test

- **Exercise**

  Write a program to reverse a string where A0 is used to point the beginning of a string and A1 is used to indicate the end of a string. Use D0 as temporary storage

- **Exercise**

  Write a program to scan a list of 10 signed number 16-bit integers . Chooses the smallest and put it in D7

- **Exercise**

  Write a program to add the absolute value of 10 signed words located in a list. The sum must be stored in a longword of D6

Self-Test

Exercise

Write a program to count the vocal characters in the string of character begins at address $5000. Place the result in D2. The NULL character is used to signify the end of the string. Calculate the byte requirement and clock cycle for the program.

Exercise

Write a program to convert to upper case character for the beginning for each word. The string begins at address $2000 and the NULL character is used to signify the end of the string.

Exercise

Write a program to count the number of upper case character in the string of character begins at address $4000. Place the result in D5. The NULL character is used to signify the end of the string. Calculate the byte requirement and clock cycle for the program.

Self-Test

- **Exercise**

  Write a program to obtain the average of 5 out of 6 quizzes. The result of the average is stored in register D0. The list of the quiz is place in location pointed by the QUIZ label.

- **Exercise**

  Write a program to count the number of string ‘BEE2223’ that are placed in the location $1000 - $2FFF. The result of the counting is stored in D3.