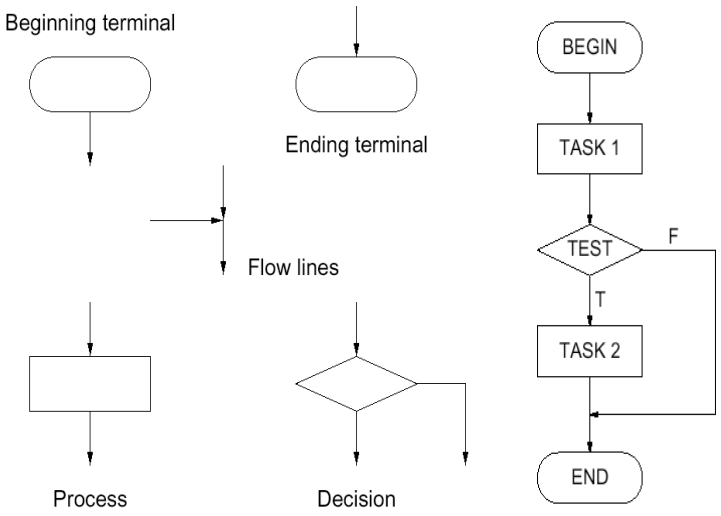
MODULAR PROGRAMMING

- Flow chart
- Software module
- Example of input/output subroutine
- Example of data processing subroutine
- Example of data changing subroutine

Flow Chart

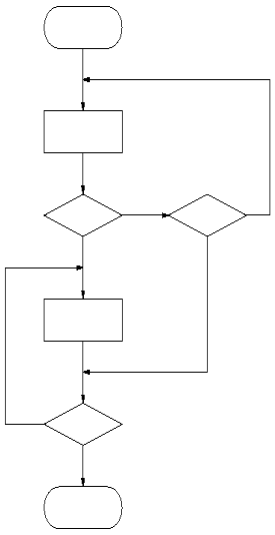
- Flow chart is a simple graphical method that can provide information a program more efficient
- Terminal – the beginning and the end of each flow chart
- Flow lines– to determine a sequence of other elements pass through
- Process – show where the work is doing
- Decision– represents the results based on a binary test, a branch for result TRUE and one more for FALSE

Flow Chart Component



The Structured Programming

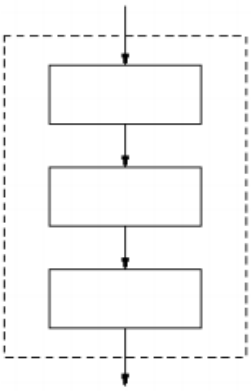
- The structured programming is a the way to write program so easy to understand and easy to modify.
- Function a program is written in a some basic structure.
- The structured programming try to avoid Laksa Johor program (spaghetti code – but we are not eat spaghetti)



Basic Constructs

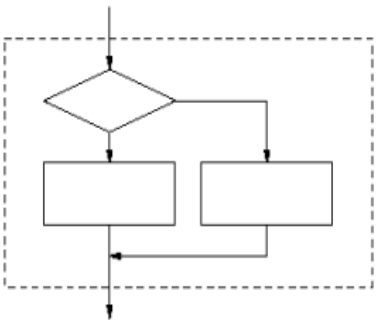
- Basic construct that has is enough to write any program!
- SEQUENCE
  a instruction that implemented in order.
- IF-THEN-ELSE
  basic decision structure
- DO-WHILE
  Basic repeat structure
- In each construct, and in the complete program structure, only have one entrance and one exit

Basic Construct - SEQUENCE



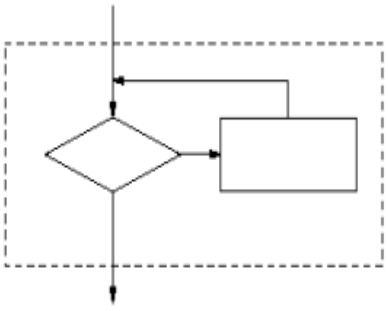
•SEQUENCE construct only one or more processes collected locally.
•Each basic structure surrounded by the dotted box to indicate the overall construct can be viewed as a process.

Basic Construct – IF-THEN-ELSE



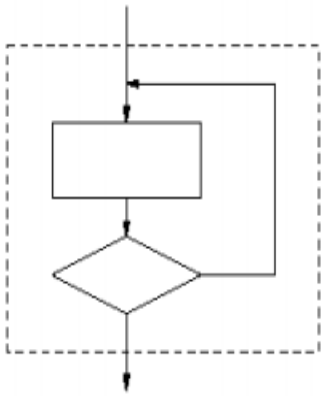
- One condition box with 2 process boxes:
- One process box is implemented if condition is TRUE and one more if condition is FALSE.

Basic Construct – DO-WHILE



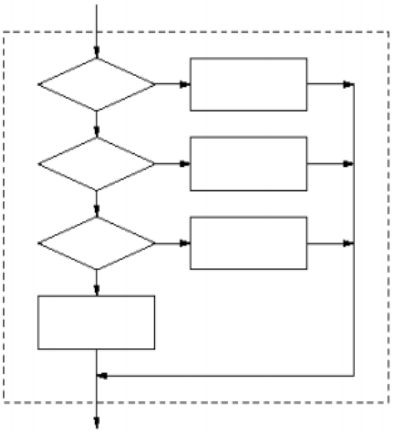
- Condition box with one process box.
- If decision TRUE, process is implemented and decision tested again.
- If FALSE, out from construct.

Additional construct –REPEAT-UNTIL



- Also known as DO-UNTIL construct
- Same as DO-WHILE but process box must implement at least once.
- Easier for assembly language than DO-WHILE.

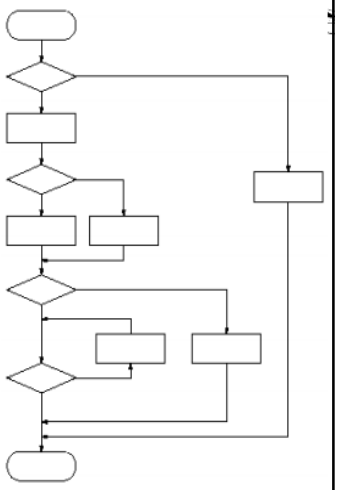
Additional Construct - CASE



- Combine many decisions with one process box for each decision.
- Can be build with combine some IF-THEN-ELSE construct.

Top Down Design

- Top down design means design a program start from high level and fill in details later.
- Flow chart can be used for this method.
- High level flow chart give the overall program flow.
- The content of process box specified with low level flow chart.



toupper

```
*TOUPPER:
*Function: convert lower case to upper up case
*          Other character not affected.
*Input:    ACCA = character that will be converted
*Output:   ACCA = character that has converted
TOUPPER CMPA    #'a
        BLO     TOU_RET
        CMPA    #'z
        BHI     TOU_RET
        SUBA    #$20
TOU_RET RTS
```

Some conversion subroutines

```
*                                   *
*convert lower nibble               *convert lower case to
*to hex ASCII.                      *upper case.
*Args: ACCA = ASCII                 *Args: ACCA = ASCII
*                                   *
HEX2ASC!CMPA #9                     LC2UC!!!CMPA!!!!#'a
!!!!!!!!BHI!!!!!ALPHA               !!!!!!!!BLO!!!!!LU_EXIT
!!!!!!!!ADDA!!!!#$30                !!!!!!!!CMPA!!!!#'z
!!!!!!!!BRA!!!!!HX_EXIT             !       BHI!!!!!LU_EXIT
ALPHA!!!ADDA!!!!#$37                !!!!!!!!SUBA!!!!#$20
HA_EXIT!RTS                         LU_EXIT RTS
```


Timing Instruction

- Implementation time:

$$T_{ex} = N * T_E$$

T_{ex} = execution time

N = the number of clock cycles

T_E = period E-clock, $1/ f_E$

$f_E = f_{XTAL} /4$

Example: for XTAL 8 MHz, frequency E-clock = 2 MHz and T_E = 500ns

Delay Subroutine

```
*caller routine
[6]!!!!!!!!!JSR!!!!!DEL500U
*all register stay except CCR
!!!!COUNT !!EQU!!!!!196     *calculation for delay 50
[3] DEL500U!PSHA
[2] !!!!!!!!!LDAA!!!!#COUNT
[2] LOOP      DECA
[3] !!!!!!!!!BNE!!!!!LOOP
[4]!!!!!!!!!PULA
[5]!!!!!!!!!RTS
```


Delay Subroutine

$$T_{ex} = N \times T_E$$

T_{ex} = implementation time

N = total of clock cycle

T_E = period cycle E, $1/ f_E$

N = 6+3+2+ COUNT(2+3) +4+5

= 20+5(COUNT)

$N = T_{ex} / T_E = 500\mu s / 500ns = 1000$

COUNT = (1000 - 20)/5 = 196

Delay Subroutine

```
*Caller routine
[6]!!!!!!!!!JSR!!!!!DEL10MS
*all register stay except CCR
!!!!COUNT !!EQU!!!!!1998        *calculation for delay 10mS
[4] DEL10MS!PSHX
[3]!!!!!!!!!LDX!!!!!#COUNT
[2] LOOP          NOP
[2]!!!!!!!!!NOP
[3]!!!!!!!!!DEY
[3]!!!!!!!!!BNE!!!!!LOOP
[5]!!!!!!!!!PULX
[5]!!!!!!!!!RTS
```


Delay Subroutine

$$N = 6 + 4 + 3 + COUNT(2 + 2 + 3 + 3) + 5 + 5$$
$$= 23 + 10(COUNT)$$

$$N = T_{ex} / T_E = 10ms/ 500ns = 20000$$

$$COUNT = (20000 - 23)/10 = 1997.7$$

Check first whether there is COUNT fix in IX!

Output string to printer

```
REGS    EQU     $1000
PORTB   EQU     4
PORTC   EQU     3

        LDX     #REGS
!!!!!!!!LDY!!!!!#STRING
LOOP!!!!TSTA
!!!!!!!!BEQ!!!!!DONE
        LDAA    0,Y
        BRCLR   PORTC,X %10000000 *     Wait printer ready
        STAA    PORTB,X
!!!!!!!!BRA     LOOP

DONE!!!!...

STRING  FCC     /Hey, this is fun!/
        FCB     0
```


Convert to ASCII hex

```
*Table method
                LDY             #ASCTBL
!!!!!!!         TAB
                ANDA    #%00001111
                ABY
                LDAA            0,Y
                use data in ACCA
ASCTBL  FCC     /0123456789ABCDEF/

*calculation method
                ANDA    #%00001111
                ADDA    #'0
                CMPA    #'9
                BLE     NOADJUS
                ADDA    #'A-'0
NOADJUS is used data in ACCA
```

Convert Case

```
*Lower to upper case
!!!!!!!!CMPA        #'a
!!!!!!!!BLT         SKIP
!!!!!!!!CMPA        #'z
        BGT         SKIP
        SUBA        #'a-'A
SKIP            use data

*Upper to lower case
!!!!!!!!CMPA        #'A
!!!!!!!!BLT         SKIP
!!!!!!!!CMPA        #'Z
        BGT         SKIP
        ADDA        #'a-'A
SKIP            use data
```


Convert from ASCII hex

```
!!!!!!!!CMPA    #'0
!!!!!!!!BLT     ERROR
!!!!!!!!CMPA    #'9
!!!!!!!!BLE     DIGIT
        CMPA    #'A
        BLT     ERROR
        CMPA    #'F
        BGT     ERROR
OK      SUBA    #'A+10
        BRA     DONE
DIGIT   SUBA    #'0
        BRA     DONE
        ...
ERROR   display message or  ask valid data
        ...
DONE    use data
```

Subroutines vs Macros

- Main difference of macros and subroutines: code size
- Every time macro is used, code repeated and add the size of machine code
- Code for subroutine only happen once in the machine code.
- Every subroutine calls, only add code for JSR or BSR instruction.