Lecture 9:
Assembly Language Example

Today’s Topics

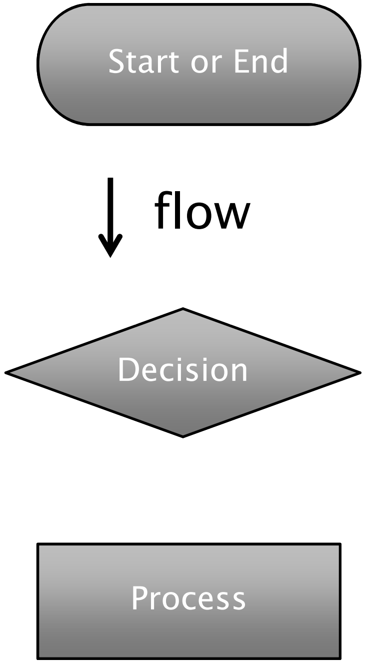
- Flowcharts
- Practice assembly programming

Flowcharts

- There are other more sophisticate methods of representing programs.
- Flowcharts work well for software written in assembly code level.
- We will see logical building blocks in flowchart formats along with assembly code templates.

Flowcharts

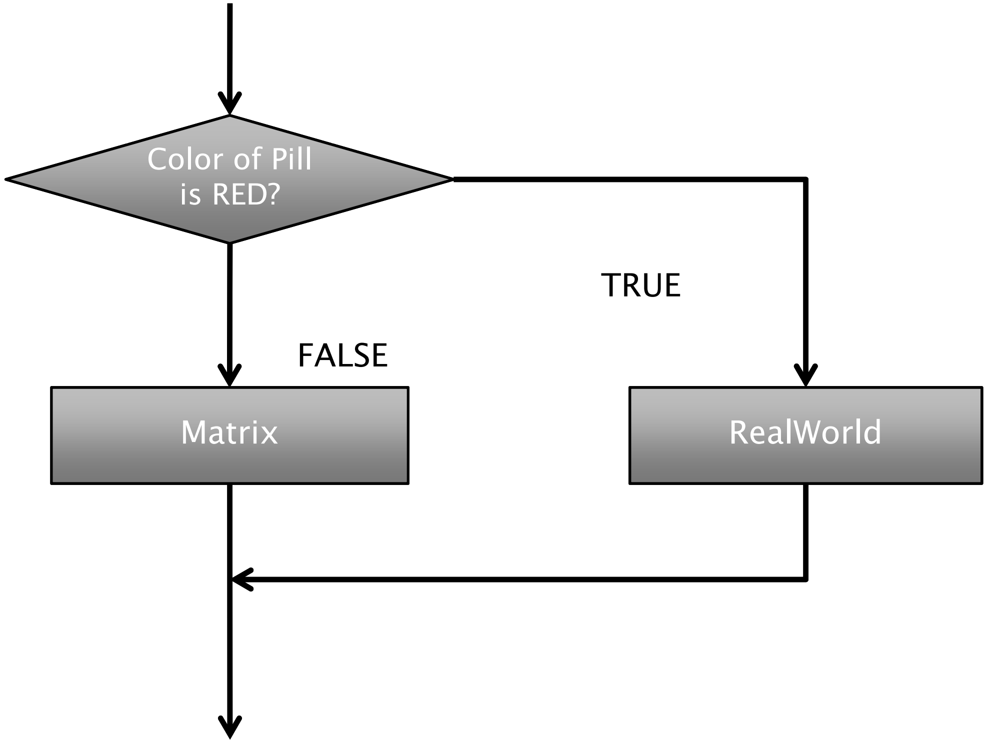
Meaning of symbols



- There are many more symbols other than these.
- Shapes may be different.
- In this course, those four symbols are pretty much all we need.

Flowcharts

If-Then-Else



Set CCR bits for decision
Bxx Process A
Process B code

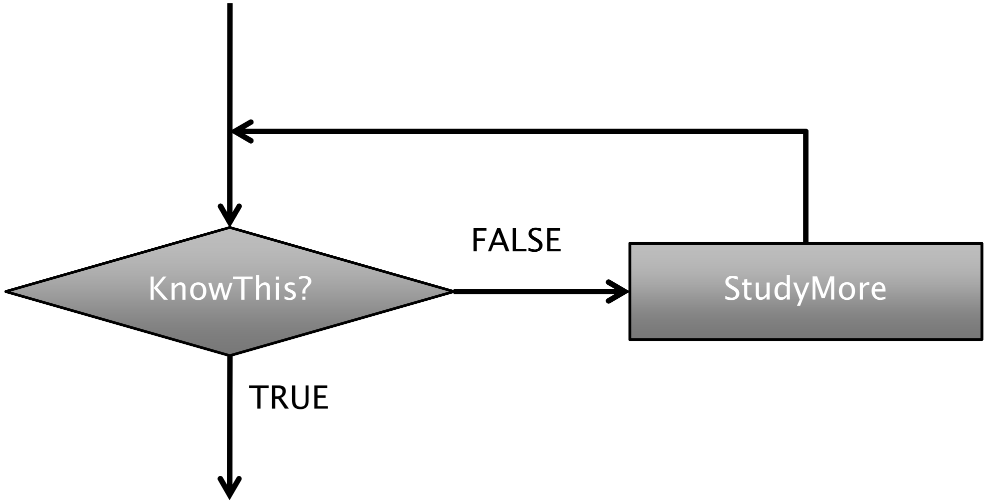
```
If(PillColor == RED)
   RealWorld
Else
   Matrix
```

Example:

```
        ldaa   PillColor
        cmpa   #RED
        bne    lmatrix
            RealWorld
            bra    lskip
lmatrix:    Matrix
lskip: ...
```

Flowcharts

While-Do



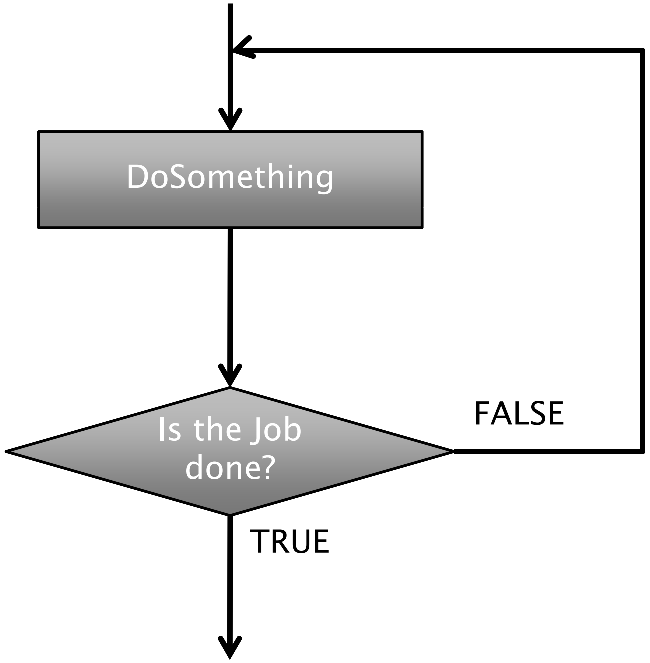
Set CCR bits for decision
Bxx past BRA
Process code
Set CCR bits for decision
BRA to Bxx

Example:

```
       ldaa   KnowThis
lloop: cmpa   #YES
       beq    lnext
       StudyMore
       ldaa   KnowThis
       bra    lloop
lnext: NextStep
```

Flowcharts

Repeat-Until



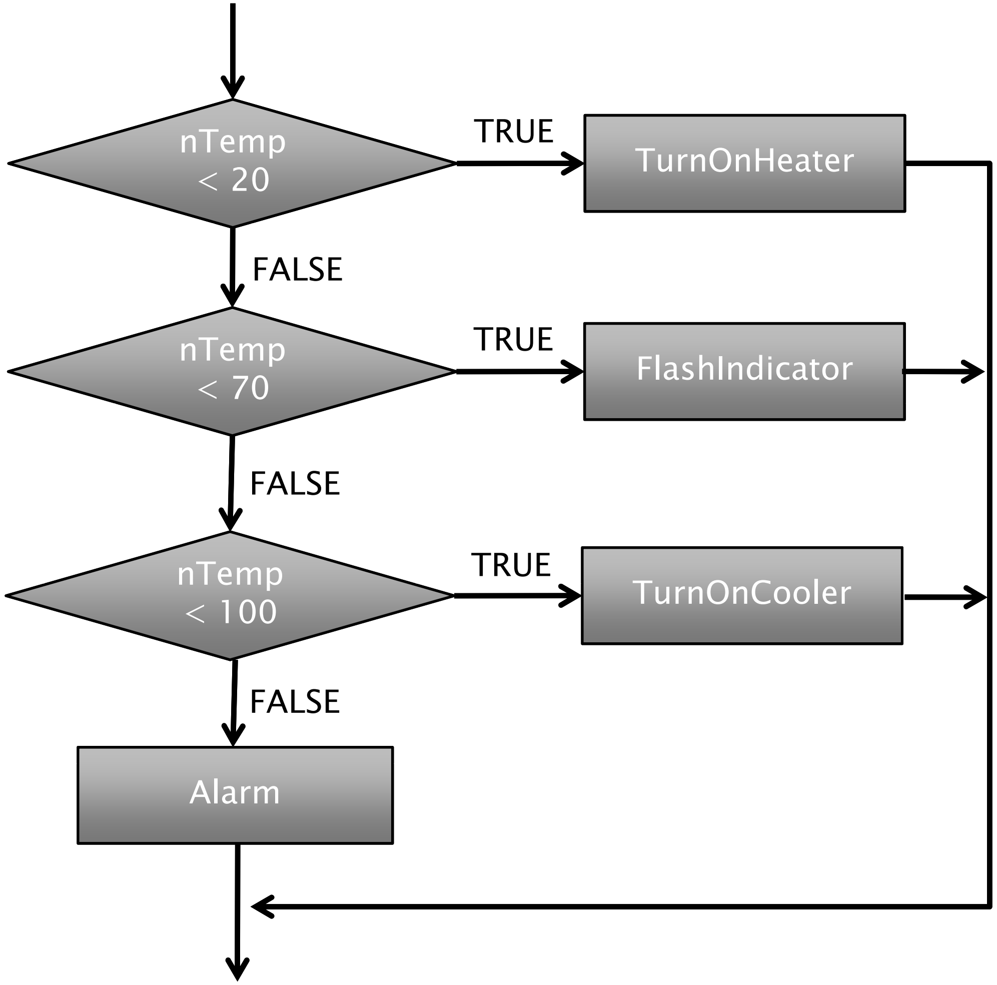
Process code
Set CCR bits
Bxx to Process code

Example:

```
loop:  DoSomething
       ldaa   Result
       beq    ldone
       bra    loop
ldone:
```

Flowcharts

Case



Set CCR bits
Bxx to Process 1
Set CCR bits
Bxx to Process 2
…
Set CCR bits
Bxx to Process N
Default Process code
BRA past Process N code
Process 1 code
BRA past process N code
Process 2 code
BRA past Process N code
…
Process N code

Example:

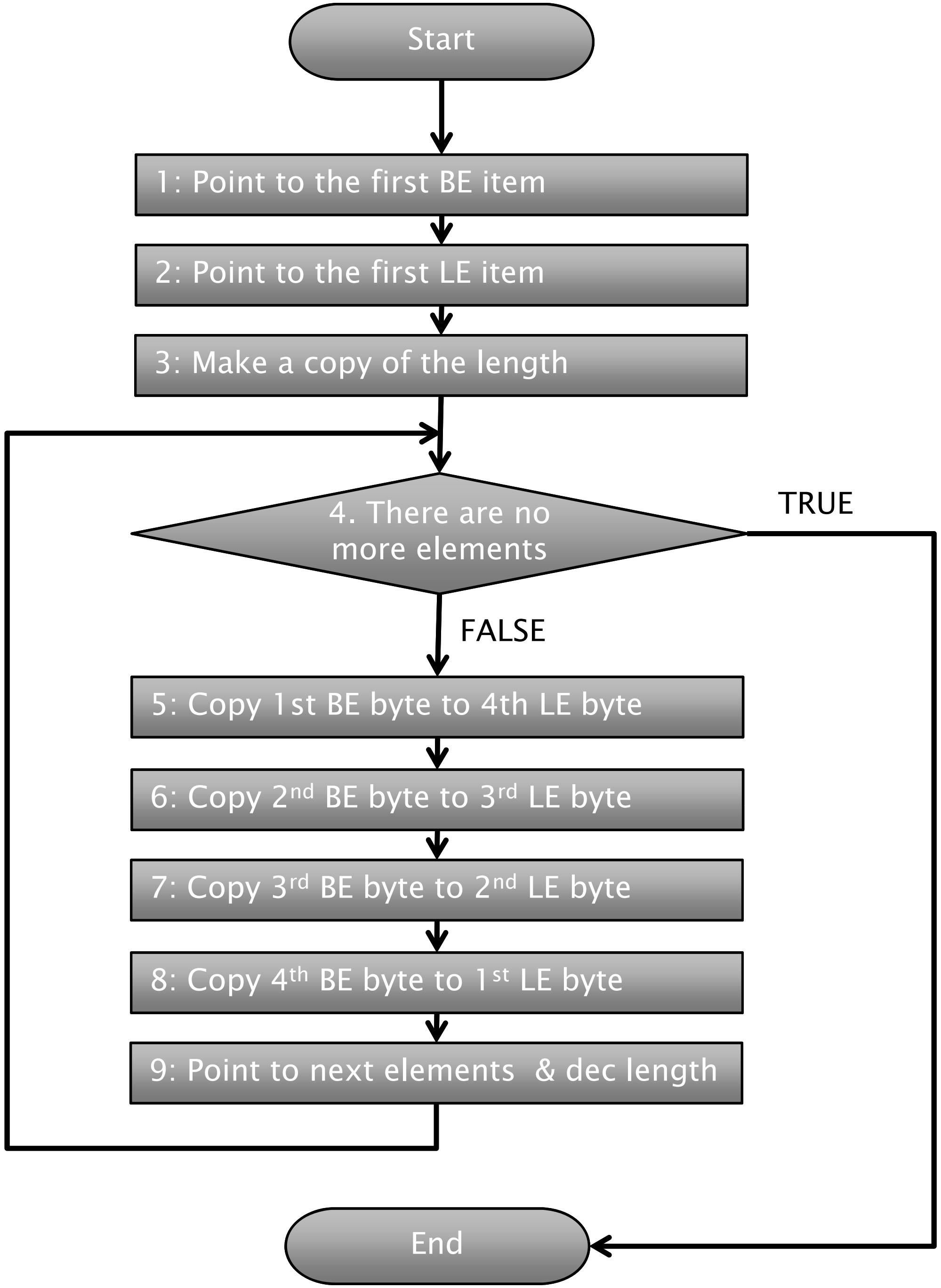
```
     ldaa nTemp
     cmpa #20
     bhs      l1
     TurnOnHeater
     bra      lres
l1:  cmpa #70
     bhi      l2
     FlashIndicator
     bra      lres
l2:       cmpa #100
     bhi      l3
     TurnOnCooler
     bra      lres
l3:  Alarm
lres:
```

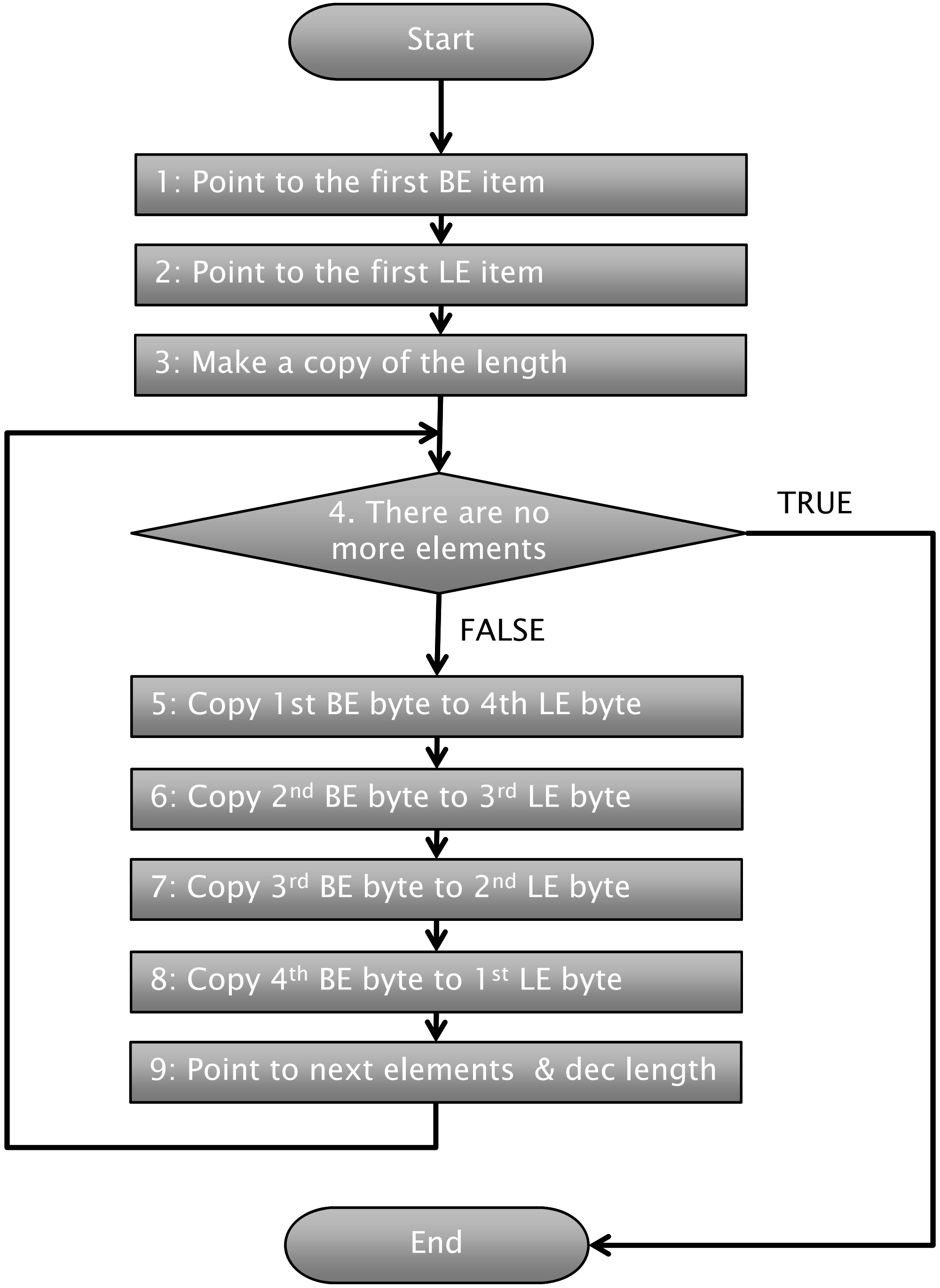
Flowchart Guidelines

- Do not refer to registers in the flowchart
- Arrows should never cross
  - They will not need to if the flowchart represents a structured program
- The purpose is to remove any questions about how to program and understand the algorithm, and this usually determines when the flowchart contains enough detail.

Assembly Example

- Convert an array of 4-byte Big-Endian values to an array of Little-Endian values.
- Let
  - $1000 hold the address of the array of Big-Endian values,
  - $1002 hold the address of the array for the Little-Endian values,
  - $1004 hold the two-byte **length** of numbers to convert.
- Write an assembly program to implement these requirements.


Start
1: Point to the first BE item
2: Point to the first LE item
3: Make a copy of the length
4. There are no more elements
TRUE
FALSE
5: Copy 1st BE byte to 4th LE byte
6: Copy 2nd BE byte to 3rd LE byte
7: Copy 3rd BE byte to 2nd LE byte
8: Copy 4th BE byte to 1st LE byte
9: Point to next elements & dec length
End



Start

1: Point to the first BE item

2: Point to the first LE item

3: Make a copy of the length

4. There are no more elements

TRUE

FALSE

5: Copy 1st BE byte to 4th LE byte

6: Copy 2nd BE byte to 3rd LE byte

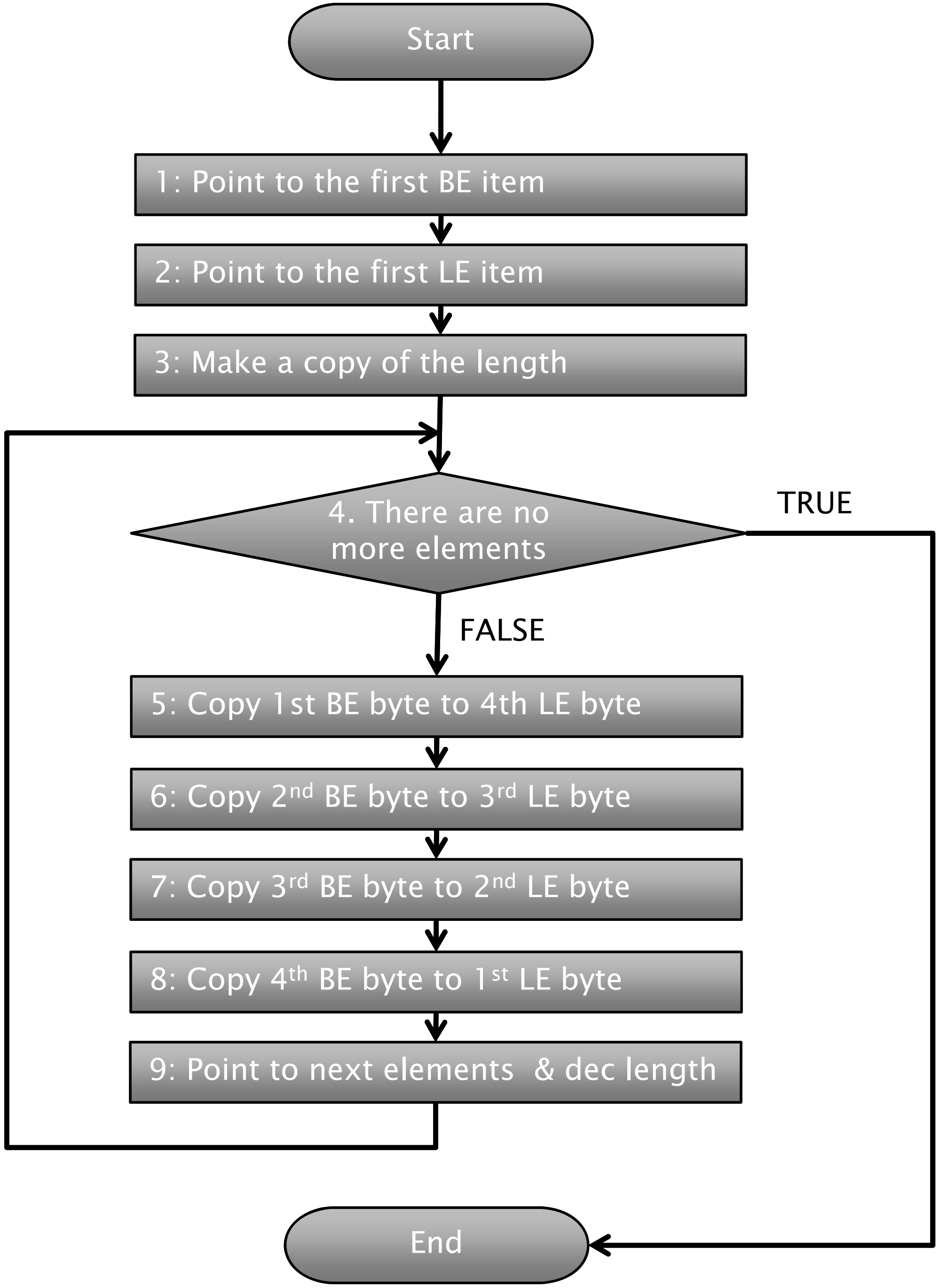
7: Copy 3rd BE byte to 2nd LE byte

8: Copy 4th BE byte to 1st LE byte

9: Point to next elements & dec length

End

```
        org     $1000
Bend    ds.w    1
LEnd    ds.w    1
Length  ds.w    1
TmpLen  ds.w    1
        org     $2000
        ldx     BEnd
        ldy     Lend
        ldd     Length
Loop    std     TmpLen   ; 3
        beq     Done
        ldaa    0,x      ; 3
        staa    3,y      ; 2
        ldaa    1,x      ; 3
        staa    2,y      ; 2
        ldaa    2,x      ; 3
        staa    1,y      ; 2
        ldaa    3,x      ; 3
        staa    0,y      ; 2
        inx              ; 1
        inx
        inx
        inx
        iny              ; 1
        iny
        iny
        iny
        ldd     TmpLen   ; 3
        subd    #$0001
        bra     Loop
Done    swi
```



Start

1: Point to the first BE item

2: Point to the first LE item

3: Make a copy of the length

4. There are no more elements

TRUE

FALSE

5: Copy 1st BE byte to 4th LE byte

6: Copy 2nd BE byte to 3rd LE byte

7: Copy 3rd BE byte to 2nd LE byte

8: Copy 4th BE byte to 1st LE byte

9: Point to next elements & dec length

End

```
        org     $1000
Bend    ds.w    1
LEnd    ds.w    1
Length  ds.w    1

        org     $2000
        ldx     BEnd
        ldy     Lend
        ldd     Length
Loop    beq     Done
        movb    0,x,3,y ; 5
        movb    1,x,2,y
        movb    2,x,1,y
        movb    3,x,0,y
        leax    4,x     ; 2
        leay    4,y
        subd    #1      ; 2
        bra     Loop
Done    swi
```

Note: You can save 10 (= 3 + 2x2 + 3) cycles.

Questions?

Wrap-up

What we’ve learned

- Flowcharts
  - Templates will be greatly helpful.
- Assembly program example

What to Come

- Arithmetic instructions
- Logic instructions