Program Control

ECE511: Digital System & Microprocessor

What we are going to learn in this session:

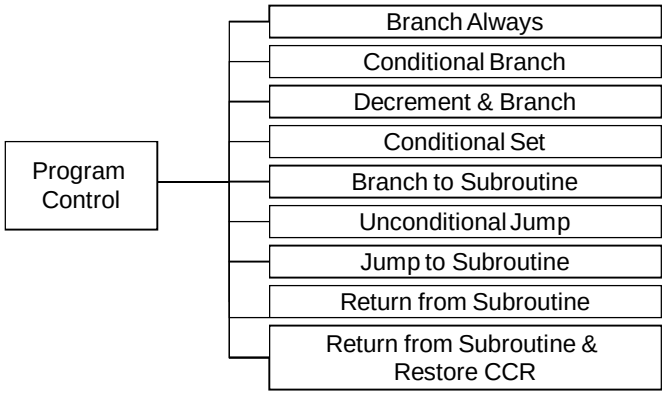
- Program Control:
  - What are they.
  - Available instructions.
  - How they effect program flow.

Introduction

Program Control Group

- Instructions that perform
  - Conditional execution.
  - Unconditional execution.
  - Subroutine handling.
  - Achieves this through branching.
    - Manipulate PC.

The Program Control Group



Unconditional Branch

BRA (Branch Always)

- Performs unconditional branching.
- Used to create infinite loops.
  - Repeats the same instructions over and over.
  - Doesn't stop until M68k resets/turned off.

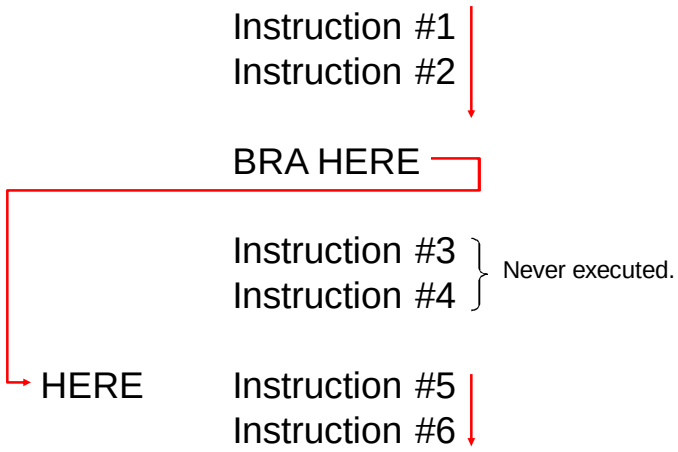
BRA (Branch Always)

- Uses relative addressing mode:
  - Has limitations.
  - 16-bit signed displacement: -32,768 to 32,767.

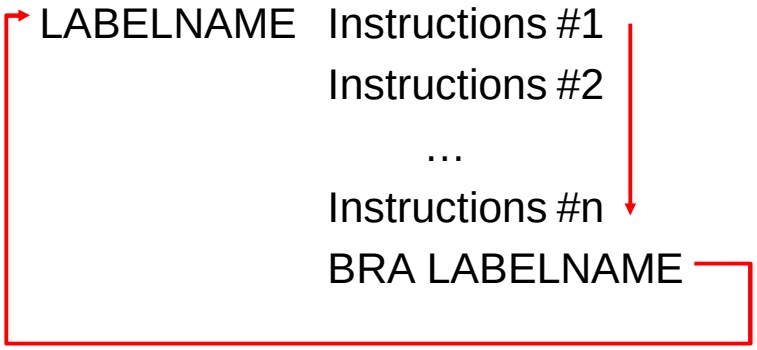
BRA Format

	Example:
BRA <ea>	BRA $001000
BRA LABELNAME	BRA THERE

Unconditional Branching Example

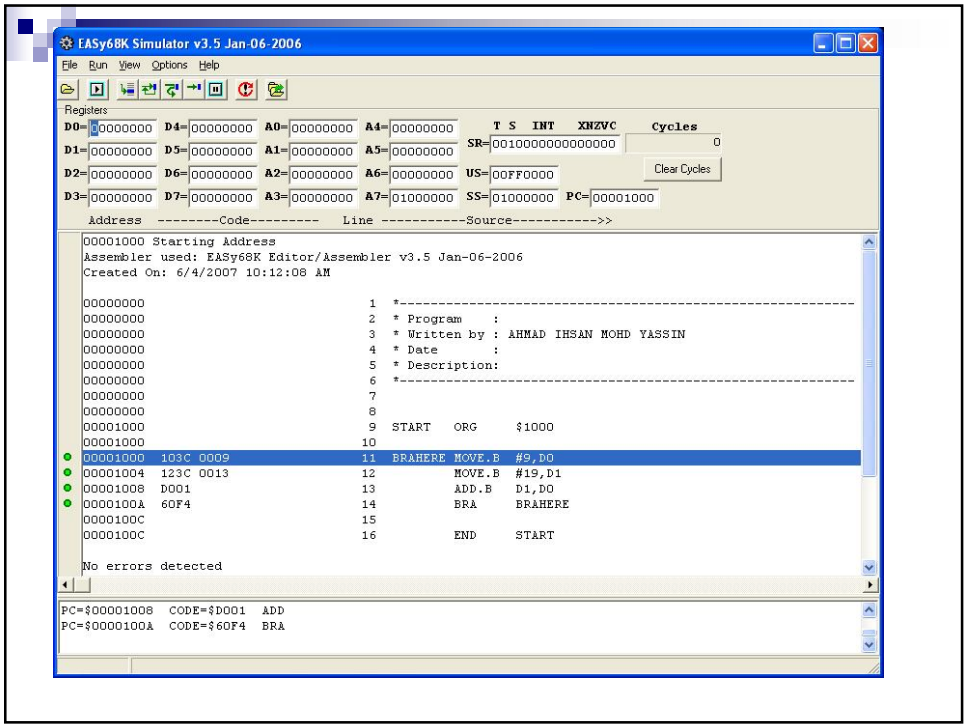


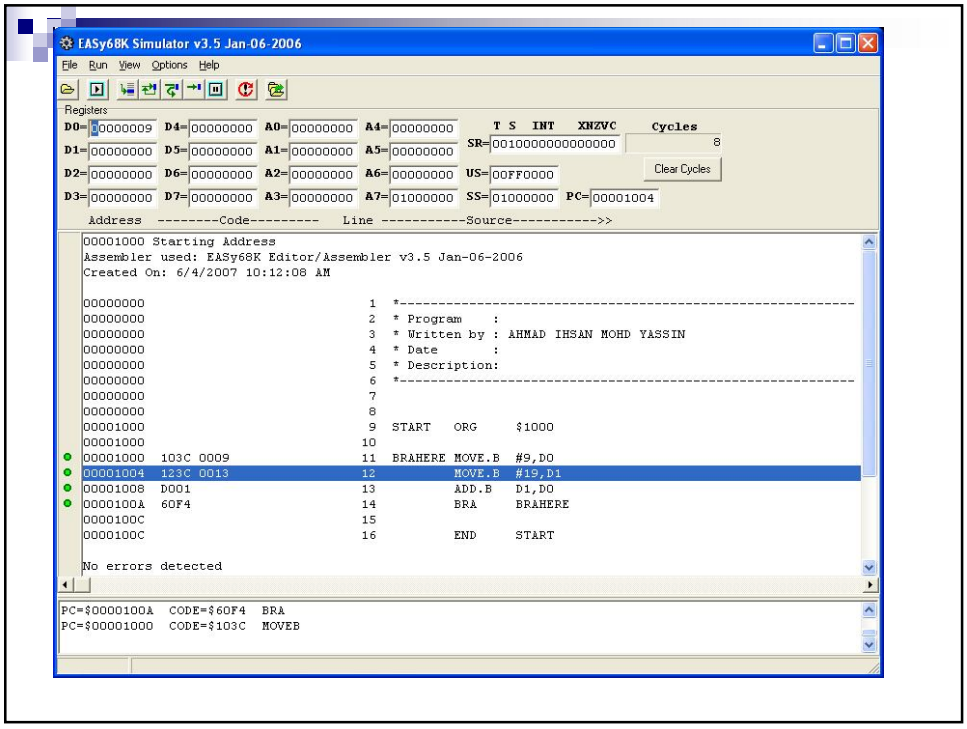
Infinite Loop Example

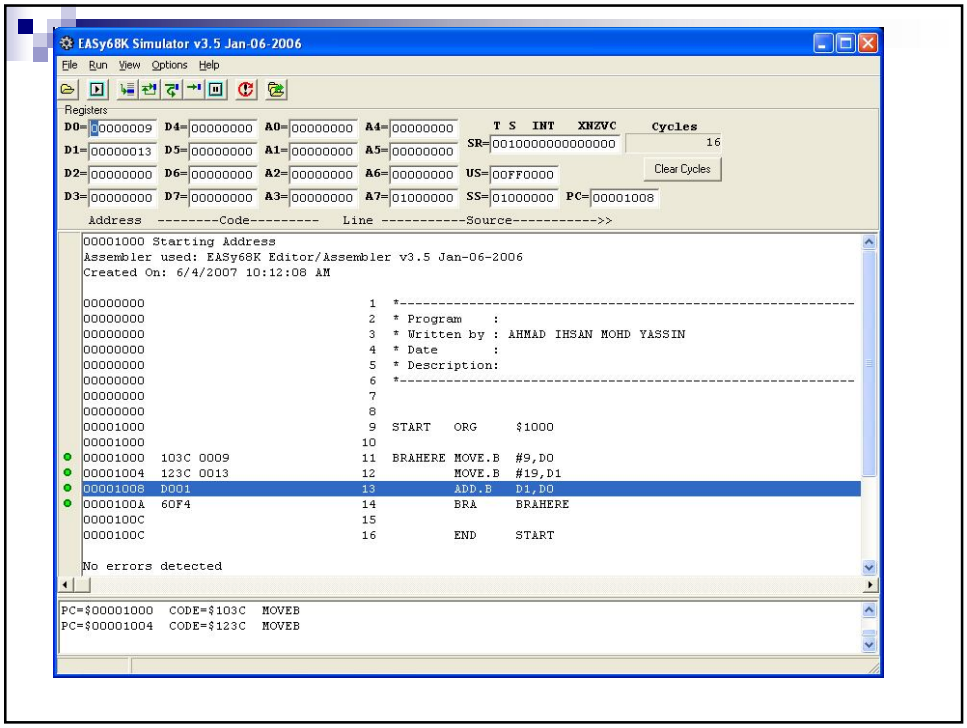


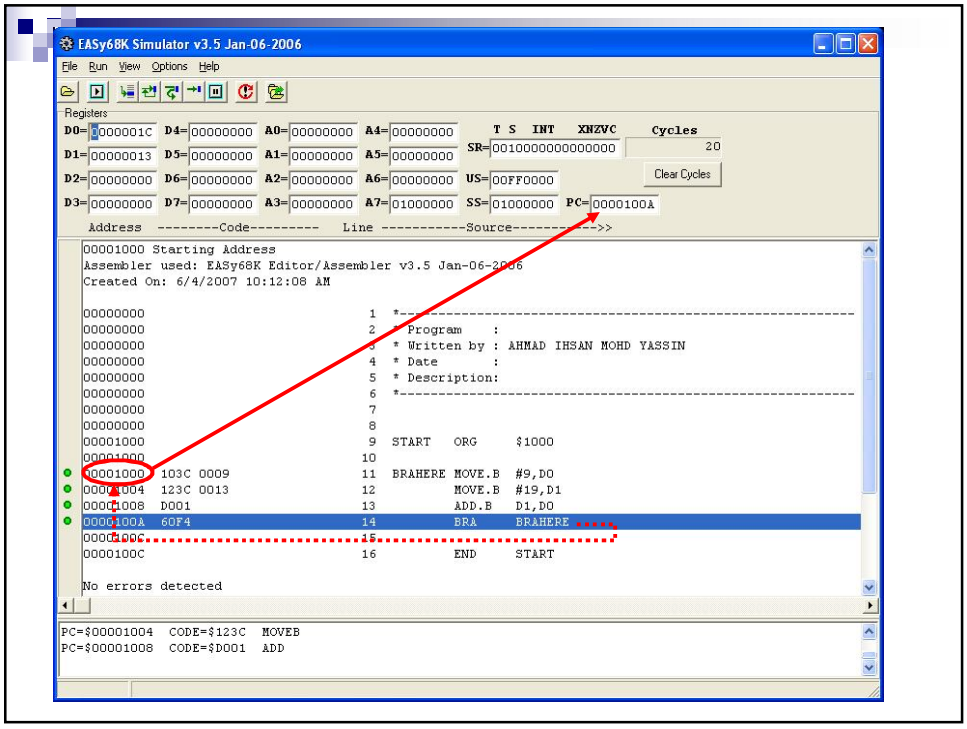
BRA Example

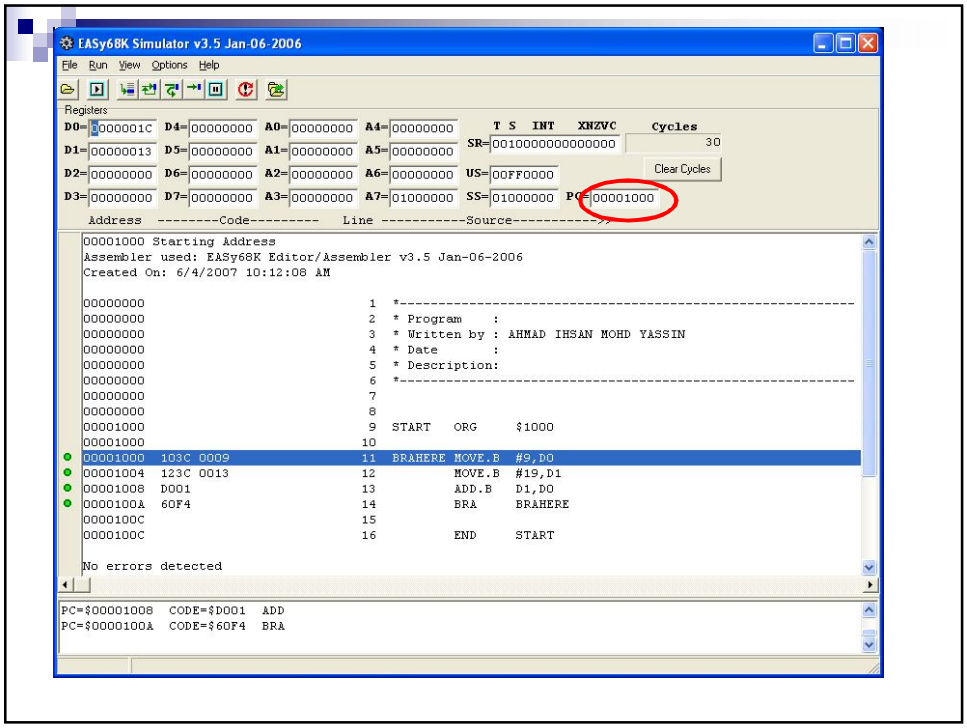
```
START       ORG     $1000

BRAHERE     MOVE.B  #9,D0
            MOVE.B  #19,D1
            ADD.B   D1,D0
            BRA     BRAHERE

            END     START
```

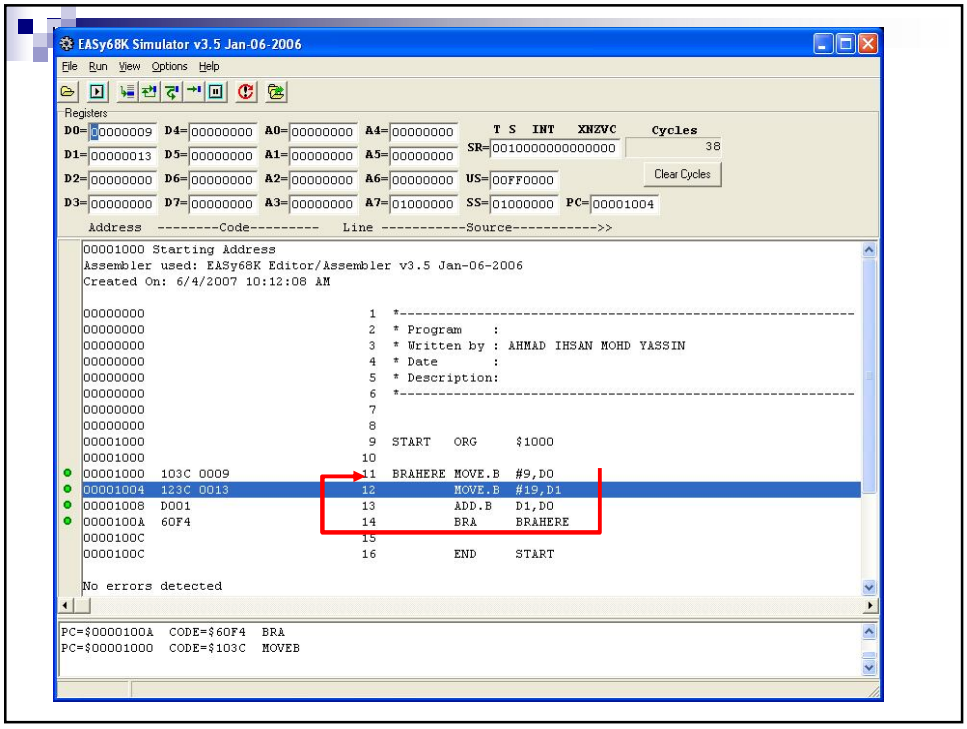

EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000000 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 0
US=00FF0000
SS=01000000 PC=00001000
Clear Cycles
Address --------Code--------- Line ----------Source----------->>
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$00001008 CODE=$D001 ADD
PC=$0000100A CODE=$60F4 BRA


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000009 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 8
US=00FF0000
SS=01000000 PC=00001004
Clear Cycles
Address --------Code--------- Line ----------Source----------->>
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$0000100A CODE=$60F4 BRA
PC=$00001000 CODE=$103C MOVEB


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000009 D4=00000000 A0=00000000 A4=00000000
D1=00000013 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 16
US=00FF0000
SS=01000000 PC=00001008
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$00001000 CODE=$103C MOVEB
PC=$00001004 CODE=$123C MOVEB


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=0000001C D4=00000000 A0=00000000 A4=00000000
D1=00000013 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 20
US=00FF0000
SS=01000000 PC=0000100A
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$00001004 CODE=$123C MOVEB
PC=$00001008 CODE=$D001 ADD


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=0000001C D4=00000000 A0=00000000 A4=00000000
D1=00000013 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 30
US=00FF0000
SS=01000000 PC=00001000
Clear Cycles
Address --------Code--------- Line ----------Source----------
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$00001008 CODE=$D001 ADD
PC=$0000100A CODE=$60F4 BRA


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000009 D4=00000000 A0=00000000 A4=00000000
D1=00000013 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 38
US=00FF0000
SS=01000000 PC=00001004
Clear Cycles
Address --------Code--------- Line ----------Source----------
00001000 Starting Address
Assembler used: EASy68K Editor/Assembler v3.5 Jan-06-2006
Created On: 6/4/2007 10:12:08 AM
00001000 103C 0009 11 BRAHERE MOVE.B #9,D0
00001004 123C 0013 12 MOVE.B #19,D1
00001008 D001 13 ADD.B D1,D0
0000100A 60F4 14 BRA BRAHERE
0000100C 15
0000100C 16 END START
No errors detected
PC=$0000100A CODE=$60F4 BRA
PC=$00001000 CODE=$103C MOVEB

BRA Limitations

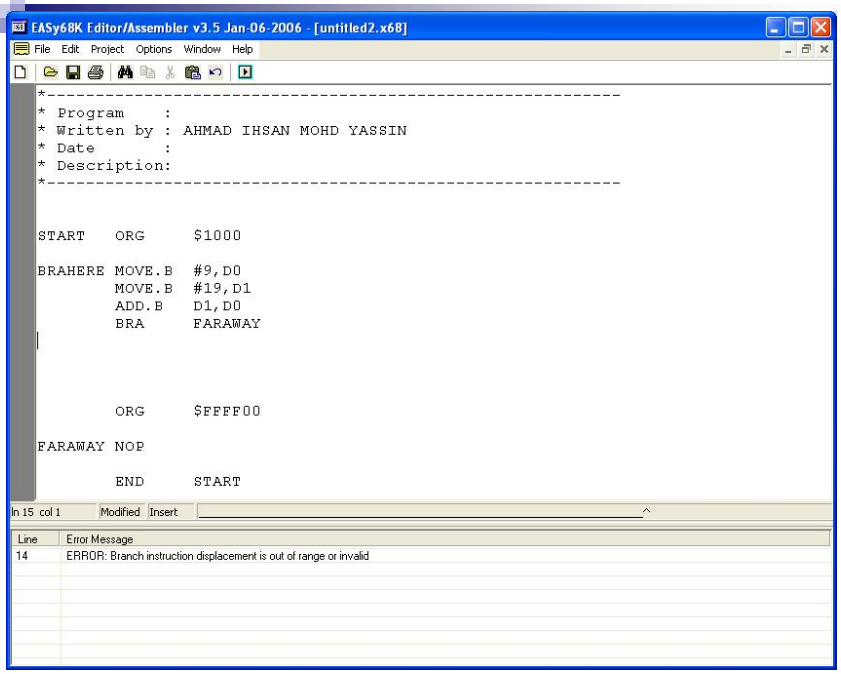
- Can address maximum 32,768 addresses back, and 32,767 addresses forward.
  - Max forward = Address + 32,767.
  - Max backward = Address – 32,768.
  - If need to be farther, can use Jump (JMP).

BRA Limitations

- Find the allowed range when BRA address is at $030000.

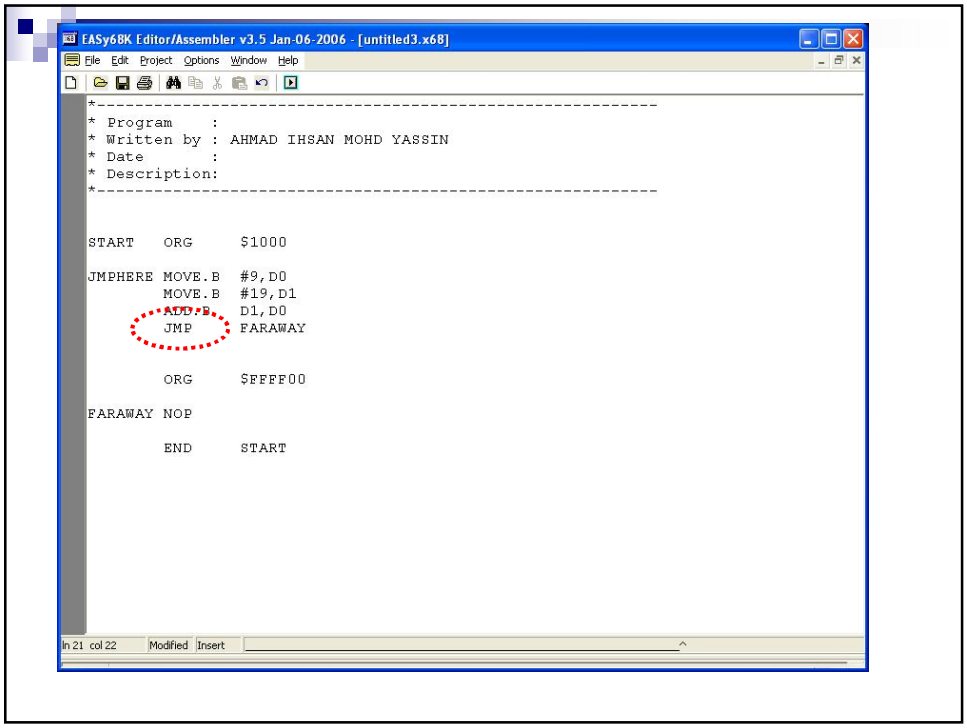
$030000 = 196,608
Max forward = 196,608 + 32,767 = 229,375 = $37FFF.
Max backward = 196,608 – 32,768 = 163,840 = $28000.

$28000 < Range < $37FFF

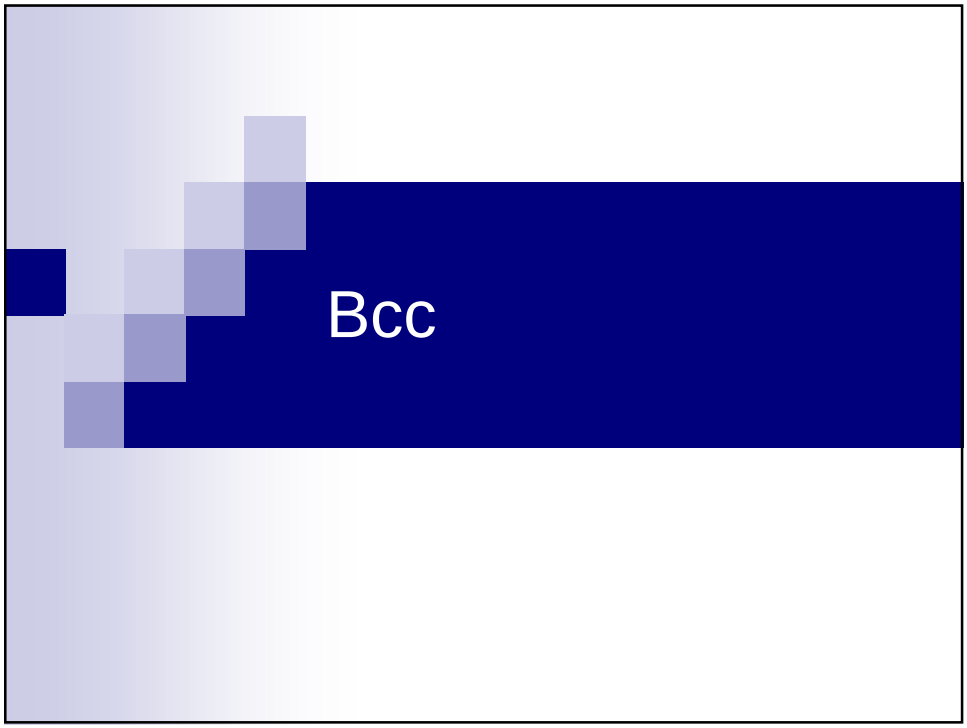


JMP (Unconditional Jump)

- Similar to BRA, but without limitations:
  - Jumps to any location in program.
  - Not limited to 16-bit displacements anymore.



Conditional Branch



Bcc (Conditional Branch)

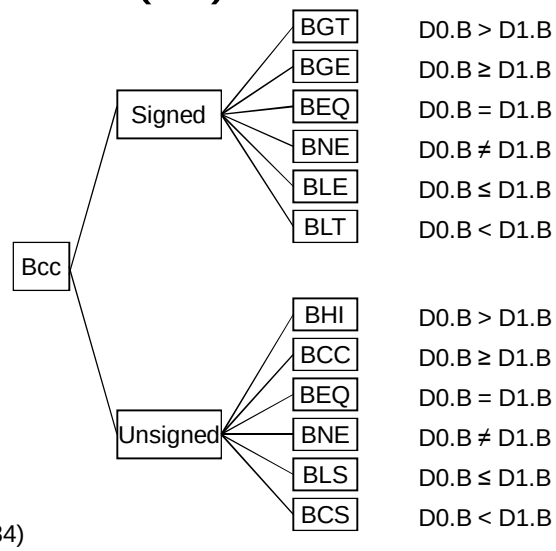
- Conditional branching:
  - Branch if condition is TRUE.
  - Execute/skip instructions results.
- Addressing limitations similar to BRA:
  - 16-bit displacement.
  - -32,768 to 32,767.

Bcc (Conditional Branch)

- Conditions tested using CMP.
  - (Antonakos, pg. 83)
  - Results stored in CCR.
- Used to create:
  - Finite loops.
  - Do…while loops.
  - If…else blocks.

Conditions (cc)

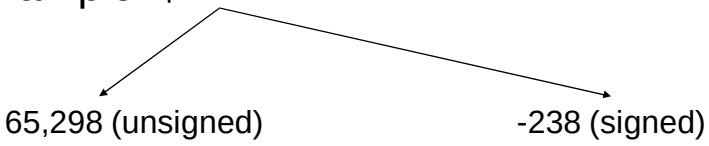
CMP.B D1,D0



*(Antonakos, pg. 84)

Why do we need signed/unsigned conditions?

- Signed/unsigned numbers carry different values.
- Example: $FF12:



- Need to choose cc properly.

Bcc Example

D0 = $000000FE
D1 = $00000023
CMP.B D1,D0

X	N	Z	V	C
-	1	0	0	0

If treated as unsigned number,

$FE = 254
$23 = 35

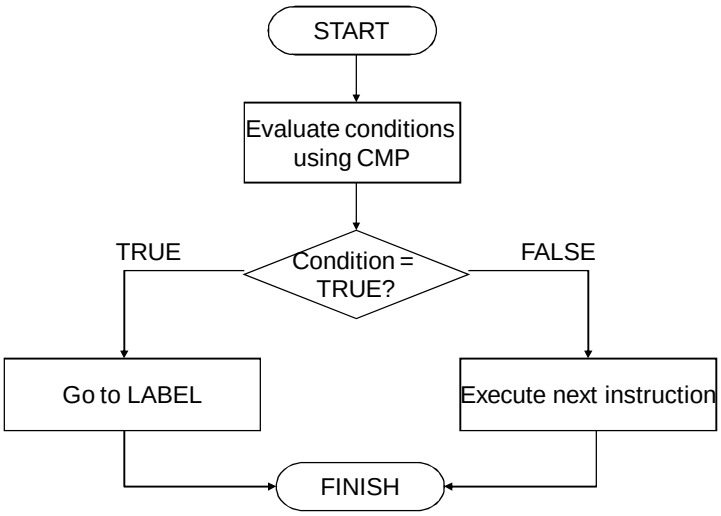
D0.B > D1.B

If treated as signed number,

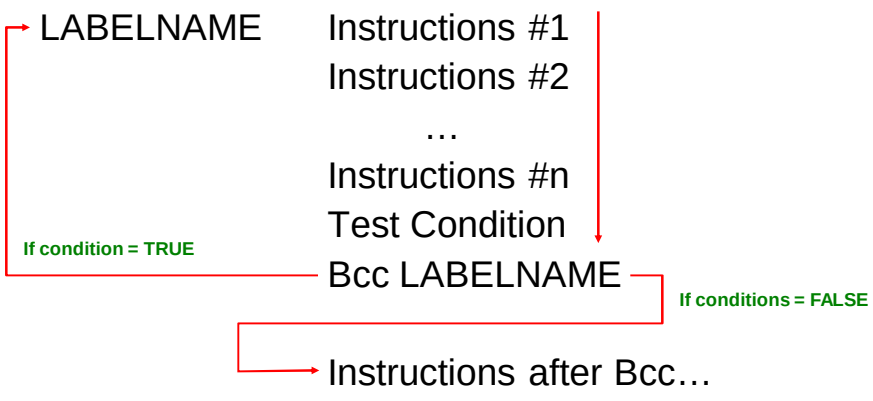
$FE = -2
$23 = 35

D0.B < D1.B

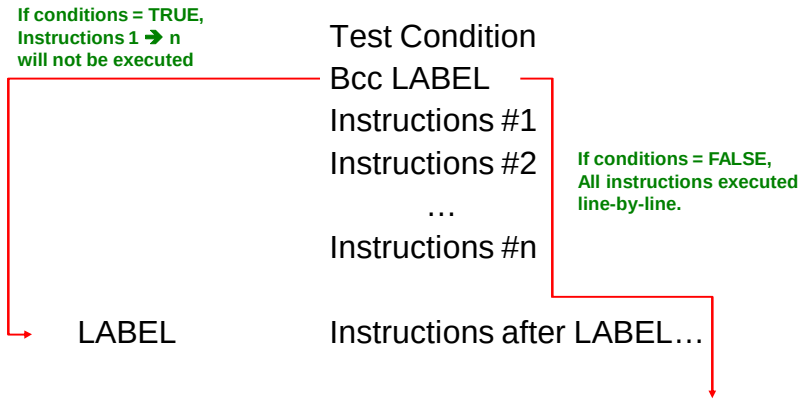
Bcc Flowchart



Bcc Format



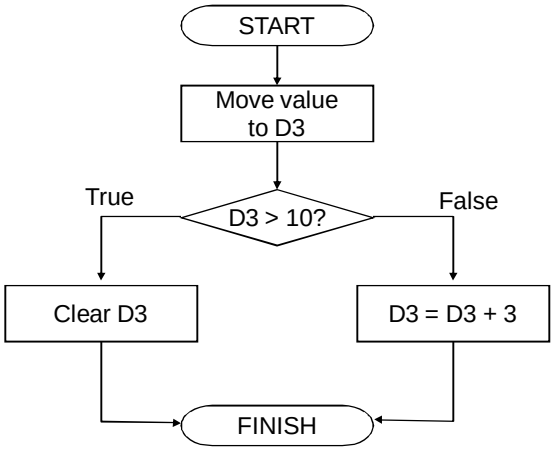
Bcc Format



If…Else Using Bcc

- Check the byte stored in D3. If D3 > 10, clear the byte. Else, add 3 to the value stored in D3.B.

Flowchart



Assembly Code

```
START    ORG     $1000
         MOVE.B  #11,D3
         CMP.B   #10,D3
         BGT     MORE
         BLE     LESS
MORE     CLR.B   D3
         BRA     FINISH
LESS     ADD.B   #3,D3
         BRA     FINISH
FINISH   END     START
```

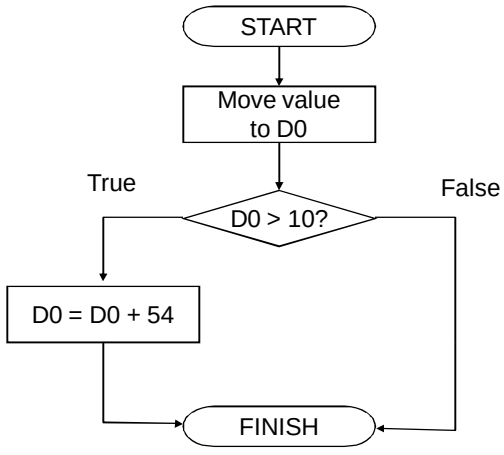
D3.B > 10

D3.B < 10

Bcc Example – If…

- Check contents of D0:
  - If D0 > 10, add 54 to it.
  - If D0 ≤ 10, do nothing.

Flowchart

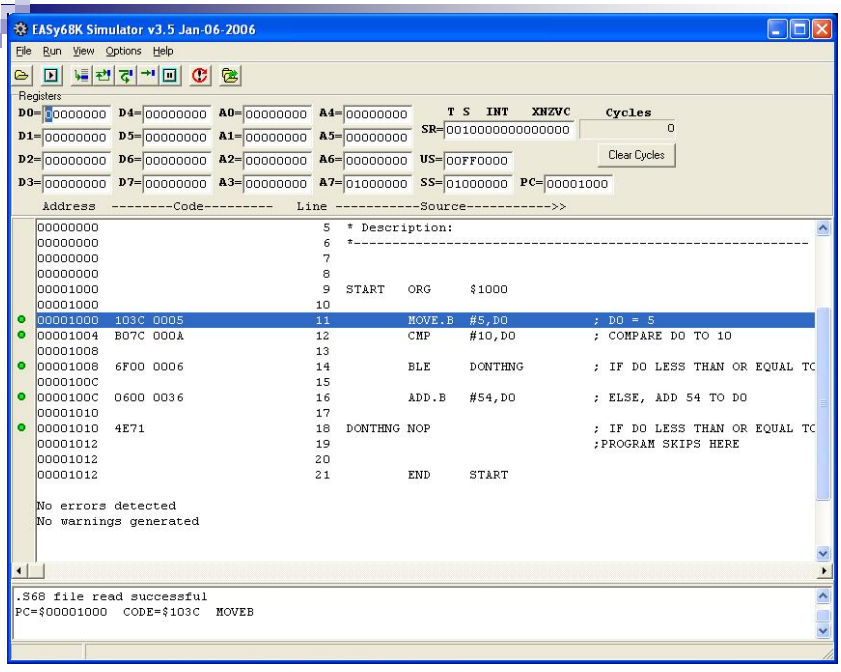


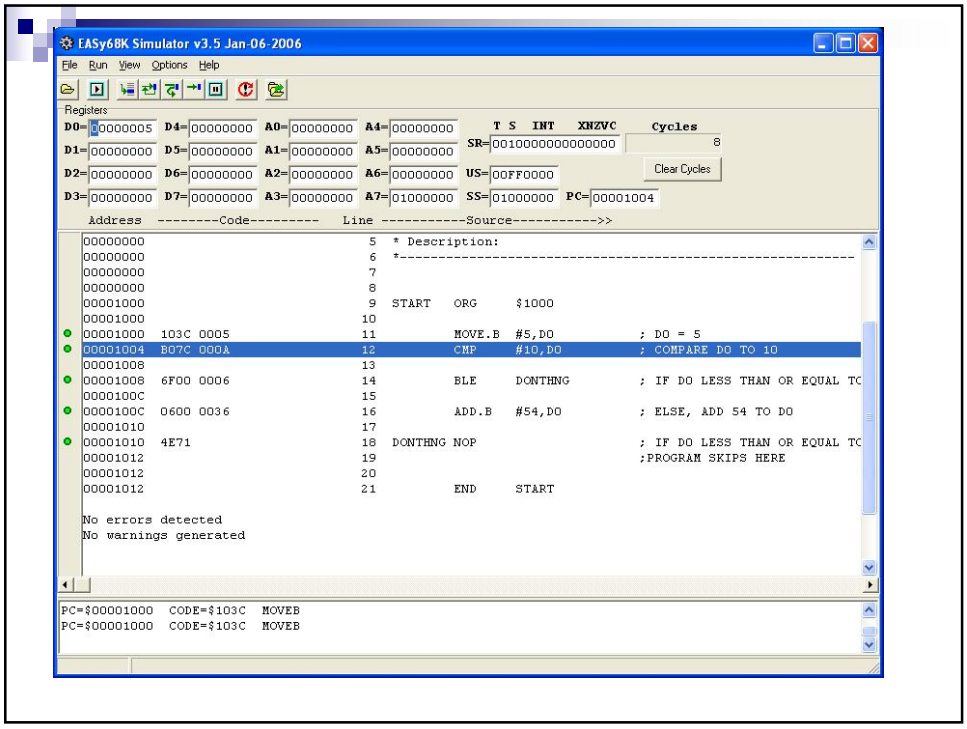
When D0 ≤ 10

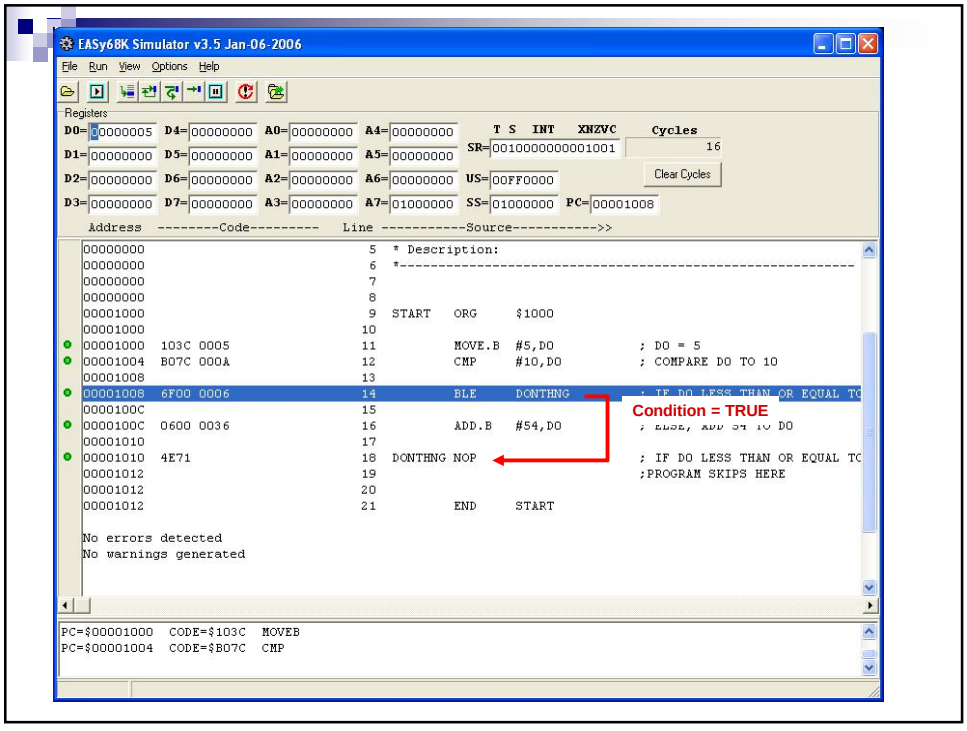
```
START     ORG       $1000

          MOVE.B    #5,D0
          CMP       #10,D0

          BLE       DONTHNG
```

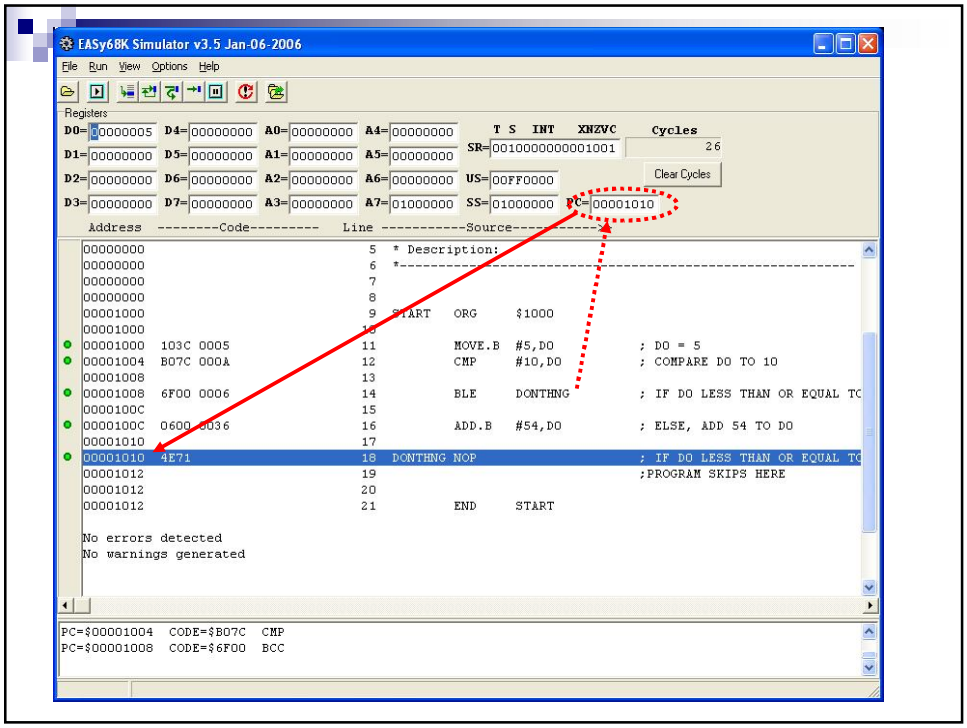
If D0 ≤ 10

```
          ADD.B     #54,D0

DONTHNG   NOP

          END       START
```



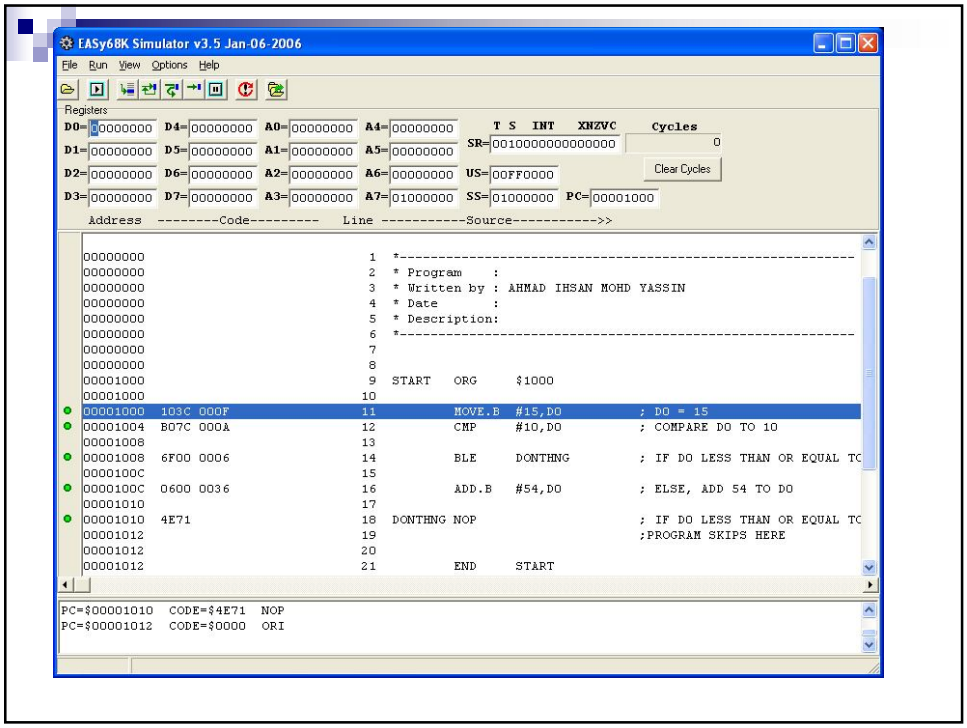

EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000005 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 8
US=00FF0000
SS=01000000 PC=00001004
Clear Cycles
Address --------Code--------- Line ----------Source----------
00000000 5 * Description:
00000000 6 *
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0005 11 MOVE.B #5,D0 ; D0 = 5
00001004 B07C 000A 12 CMP #10,D0 ; COMPARE D0 TO 10
00001008 13
00001008 6F00 0006 14 BLE DONTHNG ; IF D0 LESS THAN OR EQUAL TO
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0 ; ELSE, ADD 54 TO D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
No errors detected
No warnings generated
PC=$00001000 CODE=$103C MOVEB
PC=$00001000 CODE=$103C MOVEB

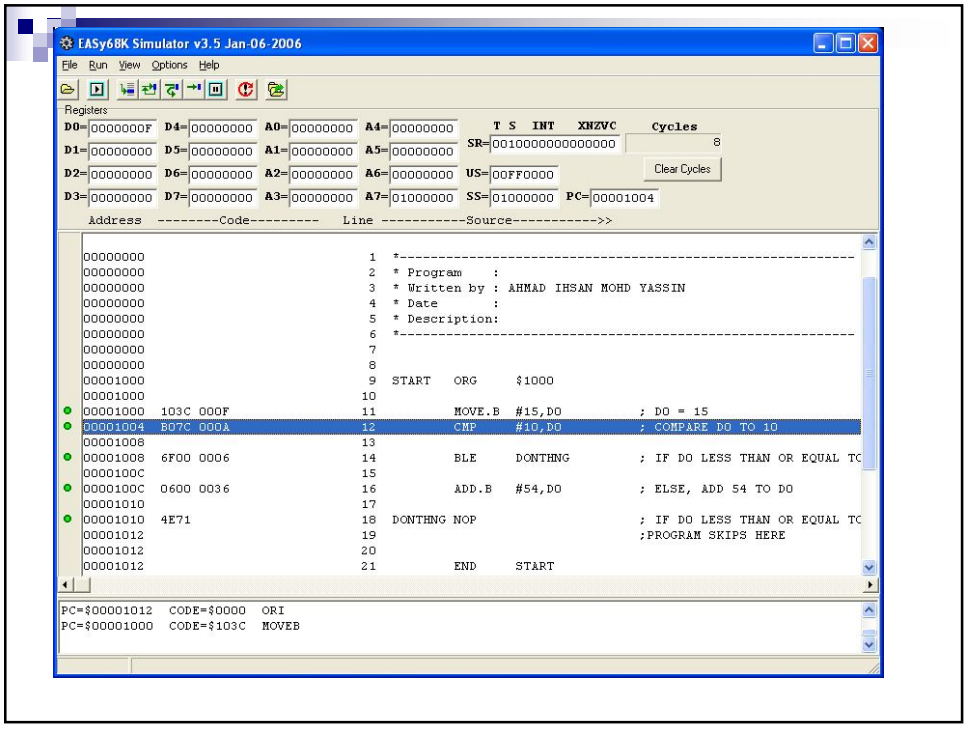

EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000005 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000001001 16
US=00FF0000
SS=01000000 PC=00001008
Clear Cycles
Address --------Code--------- Line ----------Source----------
00000000 5 * Description:
00000000 6 *
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 0005 11 MOVE.B #5,D0 ; D0 = 5
00001004 B07C 000A 12 CMP #10,D0 ; COMPARE D0 TO 10
00001008 13
00001008 6F00 0006 14 BLE DONTHNG
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
Condition = TRUE
No errors detected
No warnings generated
PC=$00001000 CODE=$103C MOVEB
PC=$00001004 CODE=$B07C CMP

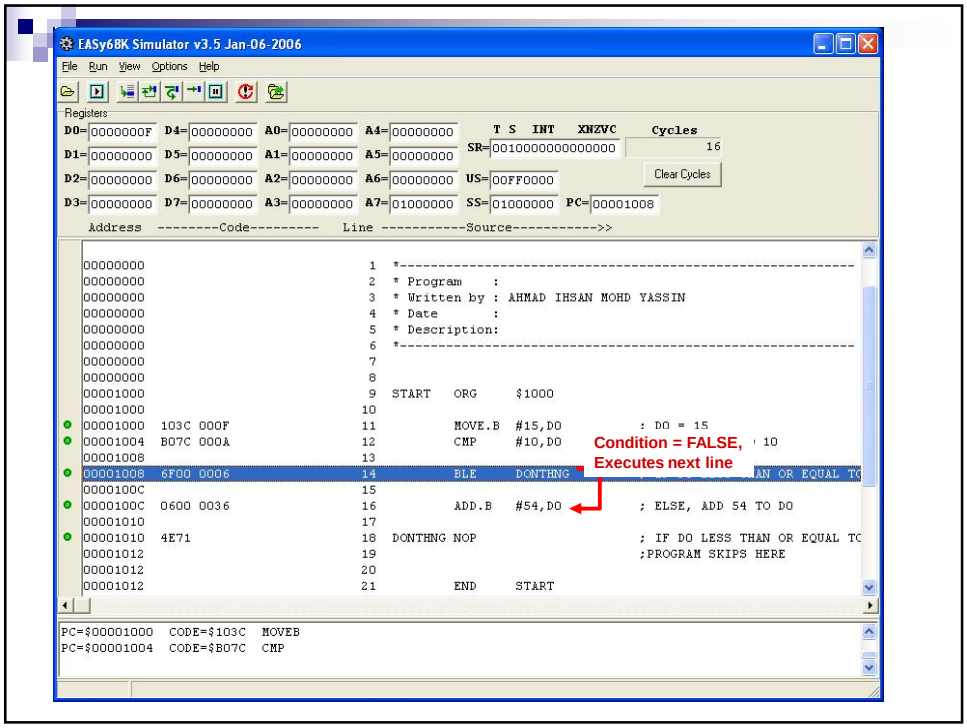


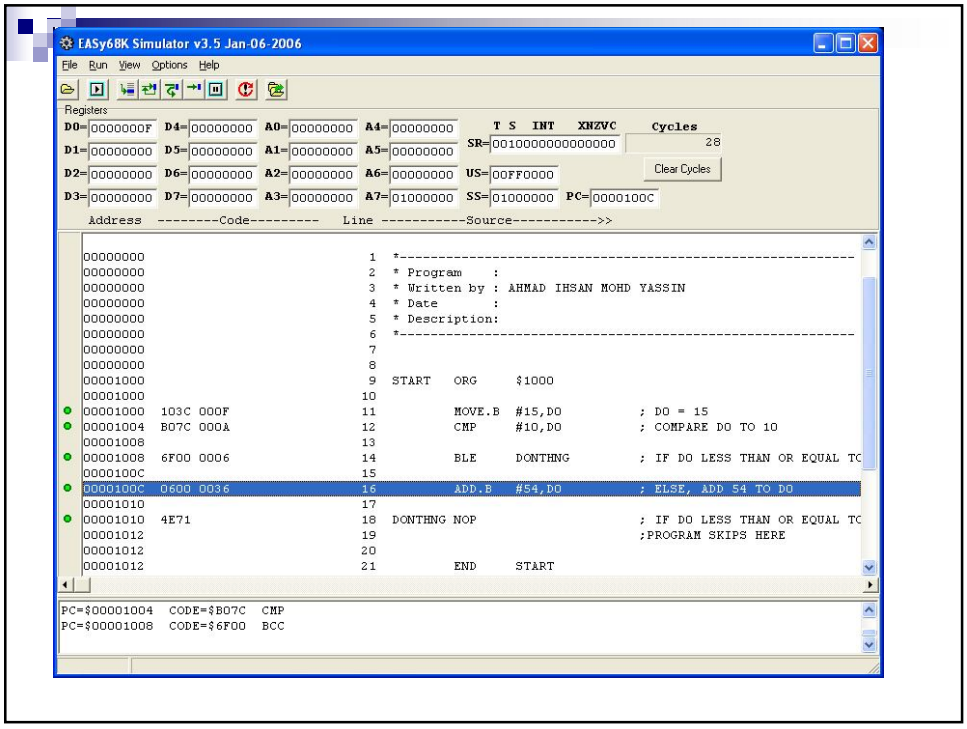
When D0 > 10

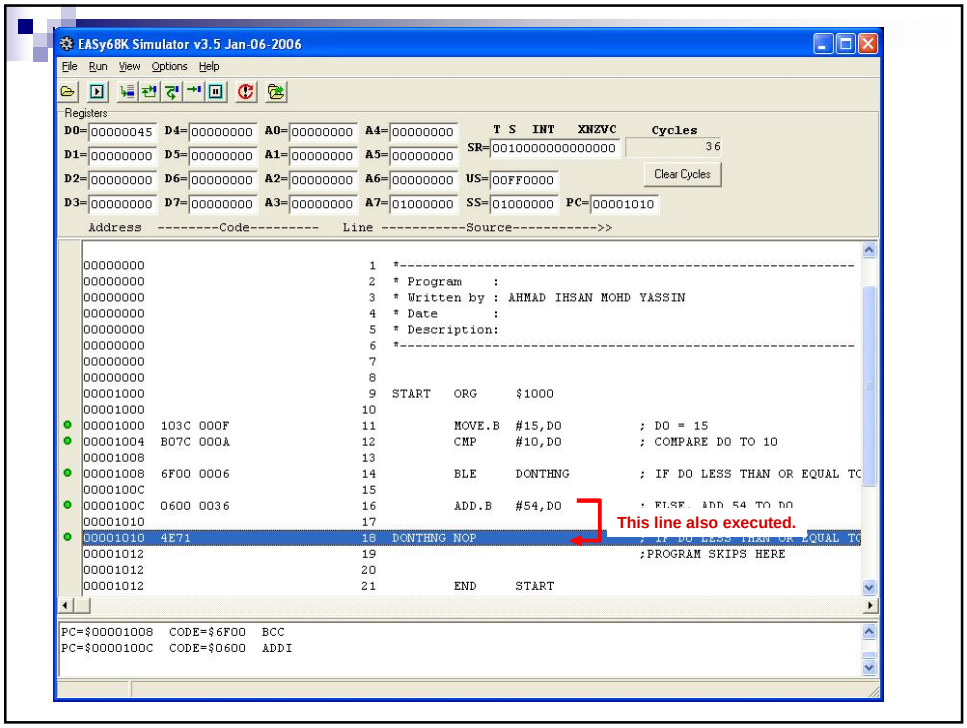
```
START       ORG         $1000

            MOVE.B      #15,D0
            CMP         #10,D0

            BLE         DONTHNG

            ADD.B       #54,D0

DONTHNG     NOP

            END         START
```


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000000 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 0
US=00FF0000
SS=01000000 PC=00001000
Clear Cycles
Address --------Code--------- Line ----------Source----------
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 000F 11 MOVE.B #15,D0 ; D0 = 15
00001004 B07C 000A 12 CMP #10,D0 ; COMPARE D0 TO 10
00001008 13
00001008 6F00 0006 14 BLE DONTHNG ; IF D0 LESS THAN OR EQUAL TO
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0 ; ELSE, ADD 54 TO D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
PC=$00001010 CODE=$4E71 NOP
PC=$00001012 CODE=$0000 ORI

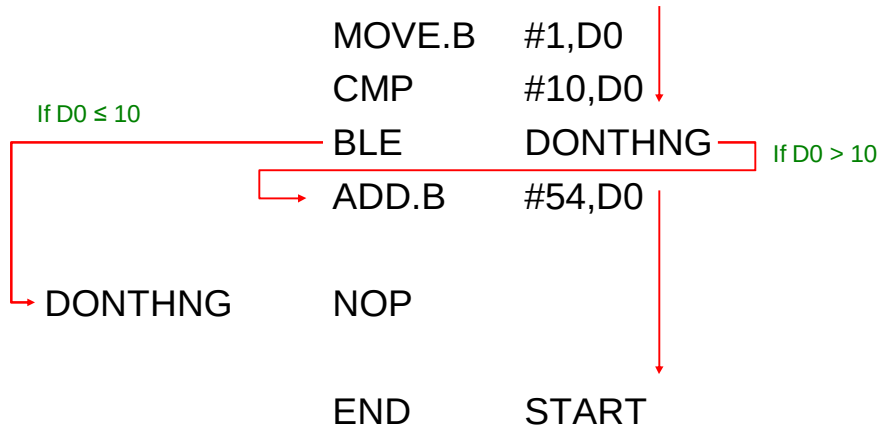

EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=0000000F D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 8
US=00FF0000
SS=01000000 PC=00001004
Clear Cycles
Address --------Code--------- Line ----------Source----------
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 000F 11 MOVE.B #15,D0 ; D0 = 15
00001004 B07C 000A 12 CMP #10,D0 ; COMPARE D0 TO 10
00001008 13
00001008 6F00 0006 14 BLE DONTHNG ; IF D0 LESS THAN OR EQUAL TO
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0 ; ELSE, ADD 54 TO D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
PC=$00001012 CODE=$0000 ORI
PC=$00001000 CODE=$103C MOVEB


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=0000000F D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 16
Clear Cycles
US=00FF0000
SS=01000000 PC=00001008
Address --------Code--------- Line ----------Source---------->>
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 000F 11 MOVE.B #15,D0 ; D0 = 15
00001004 B07C 000A 12 CMP #10,D0
00001008 13
00001008 6F00 0006 14 BLE DONTHNG
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0 ; ELSE, ADD 54 TO D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
PC=$00001000 CODE=$103C MOVEB
PC=$00001004 CODE=$B07C CMP
Condition = FALSE,
Executes next line


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=0000000F D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 28
Clear Cycles
US=00FF0000
SS=01000000 PC=0000100C
Address --------Code--------- Line ----------Source---------->>
00000000 1 *---
00000000 2 * Program :
00000000 3 * Written by : AHMAD IHSAN MOHD YASSIN
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *---
00000000 7
00000000 8
00001000 9 START ORG $1000
00001000 10
00001000 103C 000F 11 MOVE.B #15,D0 ; D0 = 15
00001004 B07C 000A 12 CMP #10,D0 ; COMPARE D0 TO 10
00001008 13
00001008 6F00 0006 14 BLE DONTHNG ; IF D0 LESS THAN OR EQUAL TO
0000100C 15
0000100C 0600 0036 16 ADD.B #54,D0 ; ELSE, ADD 54 TO D0
00001010 17
00001010 4E71 18 DONTHNG NOP ; IF D0 LESS THAN OR EQUAL TO
00001012 19 ;PROGRAM SKIPS HERE
00001012 20
00001012 21 END START
PC=$00001004 CODE=$B07C CMP
PC=$00001008 CODE=$6F00 BCC



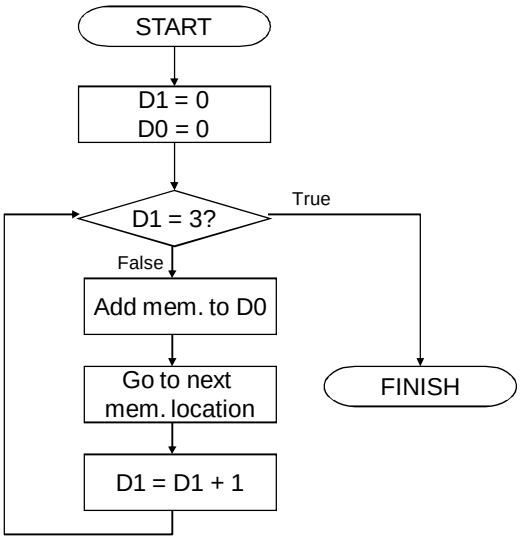
Program Flow



Bcc Example: Do..While Loop

- Add the contents of memory locations $2000, $2001, $2002 together using the do..while loop. Store the results inside D0.

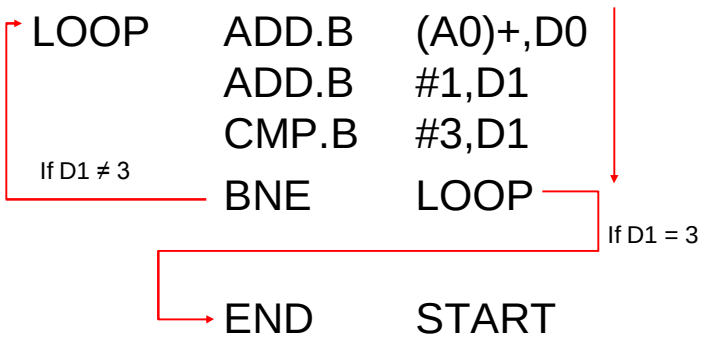
Flowchart

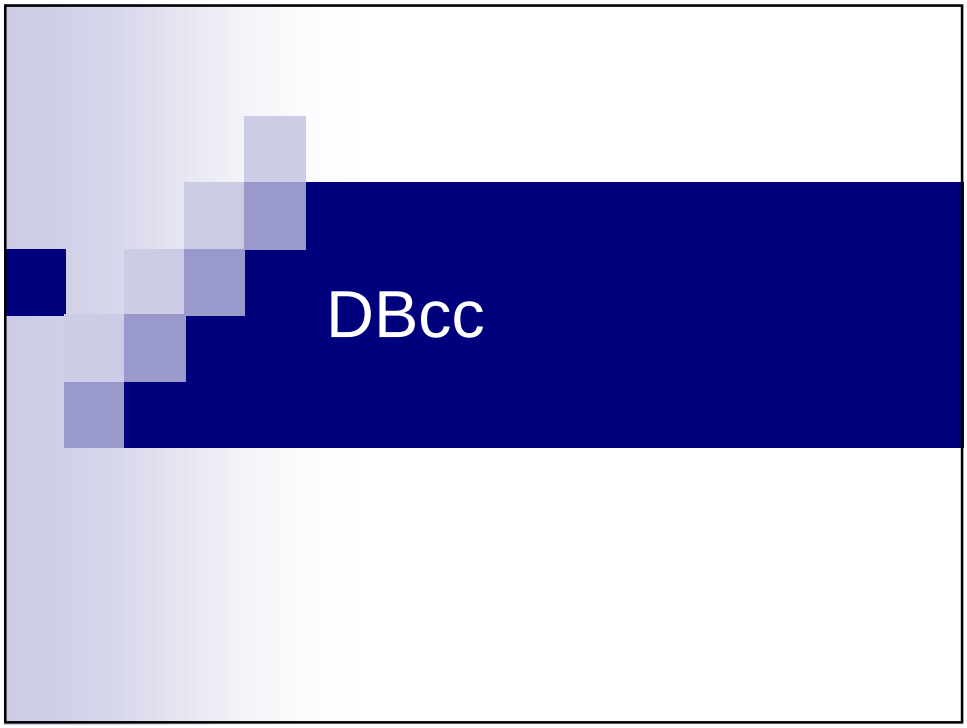


Example Program

```
START   ORG     $1000

        MOVE.B  #$12,$2000      ; PUT $12 INTO ADDRESS $2000
        MOVE.B  #$34,$2001      ; PUT $34 INTO ADDRESS $2001
        MOVE.B  #$56,$2002      ; PUT $56 INTO ADDRESS $2002

        MOVE.B  #0,D1           ; D1 AS COUNTER
        CLR.L   D0              ; D0 CLEARED, SUM STORED HERE

        LEA     $2000,A0        ; MOVE ADDRESS $2000 TO A0

LOOP    ADD.B   (A0)+,D0        ; ADD MEMORY CONTENT TO D0
                                ; INCREMENT A0 BY 1 TO POINT TO NEXT
                                ; MEM. LOCATION
        ADD.B   #1,D1           ; INCREMENT COUNTER BY 1
        CMP.B   #3,D1           ;
        BNE     LOOP            ; IF COUNTER ≠ 3, LOOP BACK

        END     START
```

Program Flow





DBcc (Test, Decrement and Branch)

- Similar to Bcc, except:
  - Now adds a counter.
  - Counter decremented at each pass.
  - Loop exits when:
    - Condition is TRUE, or
    - Counter reaches -1.

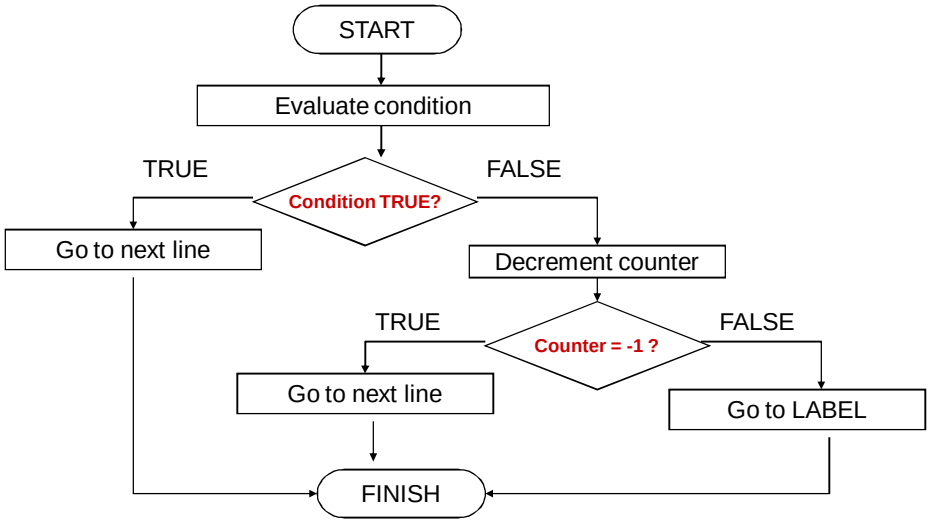
DBcc (Test, Decrement and Branch)

- Counter:
  - Word value in data register (Dn.W).
  - Decremented after each pass.
  - 0 ≤ Range ≤ 32,767.

Difference between DBcc and Bcc

	Bcc	DBcc
Branches to LABEL when..	Condition tested is TRUE	Condition tested is FALSE and when counter ≠ -1.
Go to next line when..	Condition tested is FALSE.	Condition tested is TRUE or when counter = -1.

DBcc Flowchart



Try It Yourself

```
START ORG    $1000
      MOVE.W      #10,D6
      MOVE.B      #4,D0
      CMP.B       #1,D0
      BEQ         LABEL
      MOVE.B      #$12,D4
      MOVE.B      #$34,D5

LABEL  MOVE.B     #1,D2
       END        START
```

BEQ

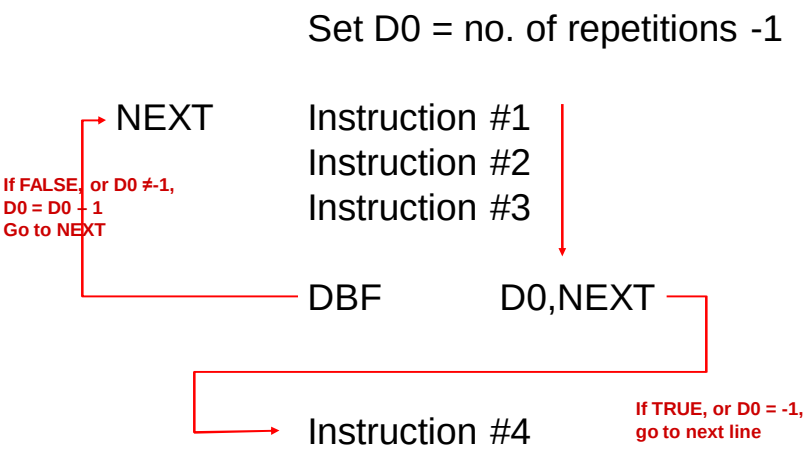
```
START ORG          $1000
      MOVE.W       #10,D6
      MOVE.B       #4,D0
      CMP.B        #1,D0
      DBEQ   D6,LABEL
      MOVE.B       #$12,D4
      MOVE.B       #$34,D5

LABEL  MOVE.B      #1,D2
       END         START
```

DBEQ

Creating for Loops using DBF

- You can create for… loops using DBF (Decrement and Branch if False).
- In DBF, result is always false:
  - Loop only determined by counter.
  - Same characteristics with for loops.
- Counter reaches -1:
  - If n loops needed, counter = n - 1.

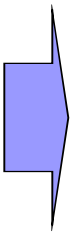
DBF Format



Example

- Write a program that fills memory locations $2000 to $2009 with value $FF.

$2000	$00
$2001	$00
$2002	$00
$2003	$00
$2004	$00
$2005	$00
$2006	$00
$2007	$00
$2008	$00
$2009	$00



$2000	$FF
$2001	$FF
$2002	$FF
$2003	$FF
$2004	$FF
$2005	$FF
$2006	$FF
$2007	$FF
$2008	$FF
$2009	$FF

Solution

- Lets say D2 is the counter.
- N = 10,
  - D2 = N – 1
  - D2 = 10 – 1
  - D2 = 9

Program

```
START    ORG      $1000

         LEA      $2000,A0
         MOVE.B   #9,D2

LOOP     MOVE.B   #$FF,(A0)+
         DBF      D2,LOOP

         END      START
```

Example

- Create a program that displays “Hello World!” 100 times.

Solution

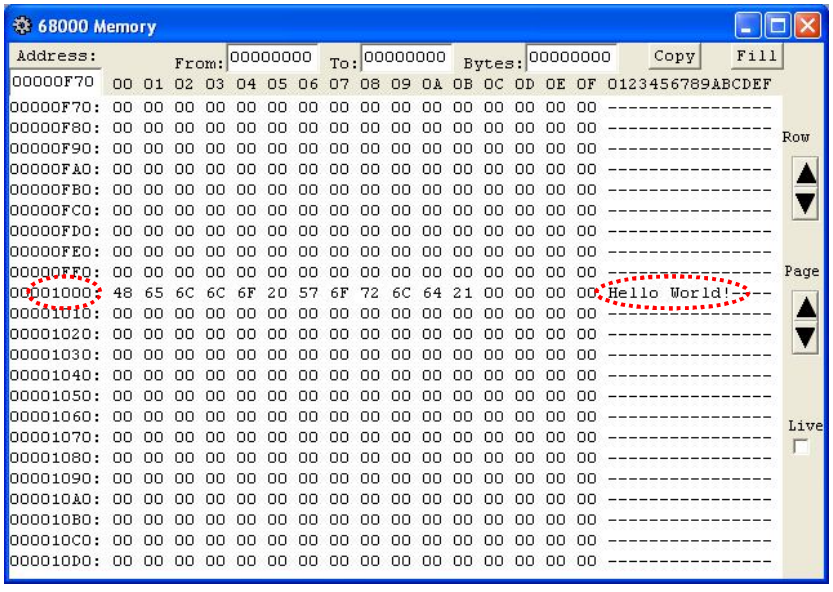
- Lets say D2 is the counter.
- N = 100,
  - D2 = N – 1
  - D2 = 100 – 1 = 99
  - D2 = $0063.

Displaying the Text

- TRAP #15 used by Easy68k for text I/O:
  - D0, D1 and A1 must be reserved:
    - D0 set to 0 to display text.
    - A1 is starting address for ‘Hello World!’ string.
    - D1 used to indicate string length: 12 characters.

Step 1: Put ‘Hello World’ into Memory

```
        ORG     $1000
HELLO   DC.B    'Hello World!'
        DC.B    0
```



Step 2: Set Variables

```
START   ORG         $2000

        MOVE.L      #0,D0
        MOVE.W      #99,D2
        MOVEA.L     #HELLO,A1
        MOVE.W      #12,D1
```

Display string mode

Counter = 99

Move string address to A1

String length

Step 3: Loop to Display Text

```
NEXT    TRAP        #15
        DBF         D2,NEXT

        END         START
```

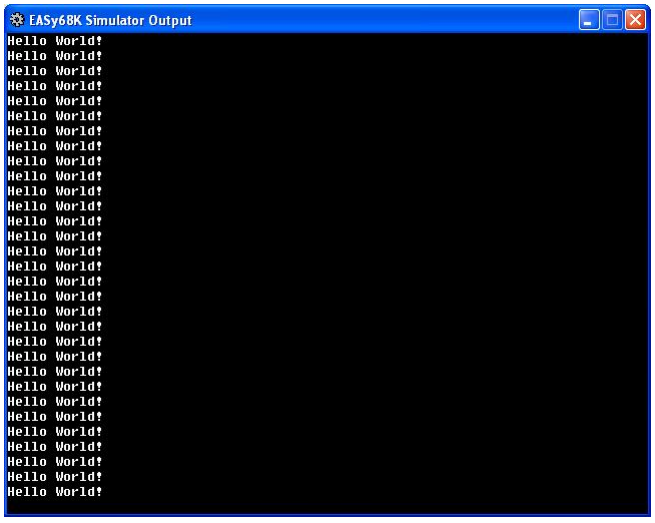
D2 = D2 -1

D2 = -1?

Assembly Code

```
        ORG             $1000
HELLO   DC.B            'Hello World!'
        DC.B            0


START   ORG             $2000

        MOVE.L          #0,D0           ; SET TO OUTPUT STRING
        MOVE.W          #99,D2          ; SET COUNTER TO 99
        MOVEA.L         #HELLO,A1       ; MOVE DC ADDRESS TO A1
        MOVE.W          #12,D1          ; THIS IS LENGTH OF CHARACTER

NEXT    TRAP            #15
        DBF             D2,NEXT         ; WILL EXIT WHEN D2 REACHES -1

        END             START
```

Result

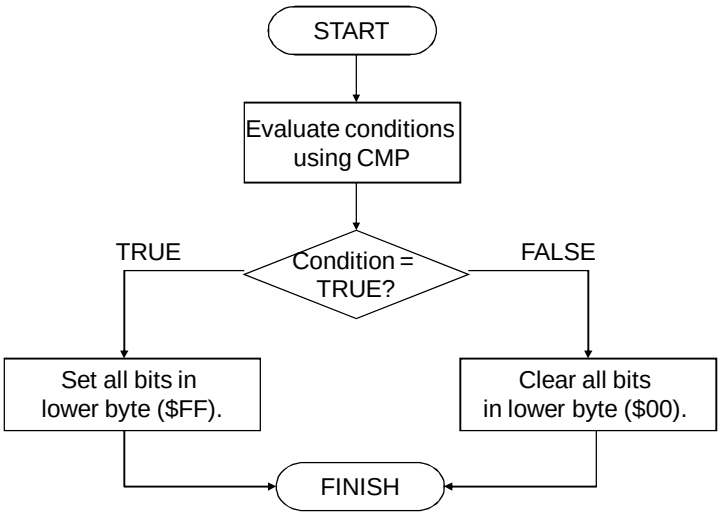


Conditional Set (Scc)

Scc (Set According to Condition)

- Check conditions and adjust byte:
  - If condition TRUE, set all bits to 1.
  - If false, set all to 0.

Scc Flowchart



Scc Example

D0 = $12345678
D1 = $12345678
CMP.B D0,D1
SEQ D2

*Original D2 = $00000000

D2.B = $FF if D1=D0, else D2.B = $00

D1.B ($78) = D0.B ($78), condition is TRUE.

D2.B =	0	0	0	0	0	0	0	0

D2.B =	1	1	1	1	1	1	1	1

Final D2 = $000000FF

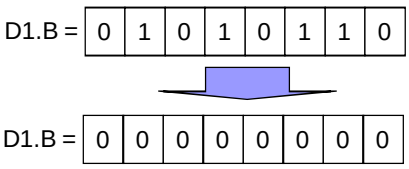
Try It Yourself

```
START    ORG      $1000

         MOVE.L   #$12345678,D0
         MOVE.L   #$12345678,D1
         CMP.B    D0,D1
         SEQ      D2


         END      START
```

Scc Example

D0 = $12345678
D1 = $10000056
CMP.B D0,D1
SGT D1 D1.B = $FF if D1>D0, else D1.B = $00

D1.B ($56) < D0.B ($78), condition is FALSE.



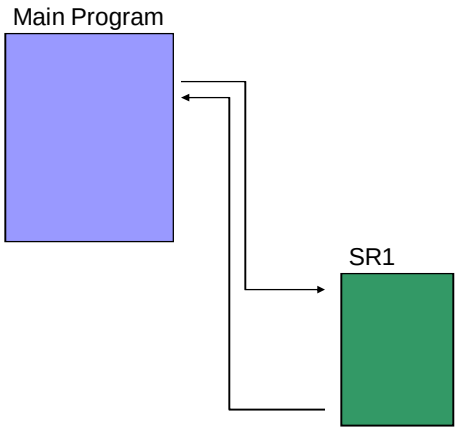
Final D1 = $10000000

Subroutine Control

Branch to Subroutine (BSR)

- Similarities to BRA:
  - Branches unconditionally to another location.
  - Same limitations (32,676 forward/back).
- Differences:
  - Location is a sub-routine.
  - *Return address* pushed to stack.
  - Subroutine returns using RTS.

Subroutines



Jump to Subroutine (JSR)

- Similar to BSR, but without addressing limit.
  - Can jump anywhere.
- Features:
  - The location is a sub-routine.
  - *Return address* pushed to stack.
  - Subroutine returns using RTS.

RTS (Return from Subroutine)

- Used to return from subroutine.
- Pops return address from stack.
- Complements BSR & JSR:
  - Put at end of subroutine.
  - Reloads PC from stack.
  - Execution resumes at new PC.

How BSR works

MAIN PROGRAM

$1000	MOVE.B #100,D0
$1002	BSR SUBR
$1004	MOVE.B D0,D1

1. Program executes normally until BSR is encountered.

PC	$1004

STACK POINTER

$FFFB	
$FFFD	
$FFFF	

SP

SUBROUTINE SUBR

$2000	MULU D0,D0
$2002	RTS
$2004	

How BSR works

MAIN PROGRAM

$1000	MOVE.B #100,D0
$1002	BSR SUBR
$1004	MOVE.B D0,D1

PC	$2000

3. Branches to SUBR by updating PC.

2. BSR saves current PC onto stack.

STACK POINTER

SP

$FFFB	
$FFFD	$001004
$FFFF	

SUBROUTINE SUBR

$2000	MULU D0,D0
$2002	RTS
$2004	

How BSR works

MAIN PROGRAM

$1000	MOVE.B #100,D0
$1002	BSR SUBR
$1004	MOVE.B D0,D1

PC	$2004

4. Execution resumes here, until encounters RTS.

STACK POINTER

SP

$FFFB	
$FFFD	$001004
$FFFF	

SUBROUTINE SUBR

$2000	MULU D0,D0
$2002	RTS
$2004	

How BSR works

MAIN PROGRAM

$1000	MOVE.B #100,D0
$1002	BSR SUBR
$1004	MOVE.B D0,D1

PC	$1004

5. When RTS is encountered,
M68k pops the stack and loads into PC.

STACK POINTER

SP

$FFFB	
$FFFD	$001004
$FFFF	

SUBROUTINE SUBR

$2000	MULU D0,D0
$2002	RTS
$2004	

How BSR works

MAIN PROGRAM

$1000	MOVE.B #100,D0
$1002	BSR SUBR
$1004	MOVE.B D0,D1

PC	$1004

6. Execution continues
normally where BSR left off.

STACK POINTER

SP

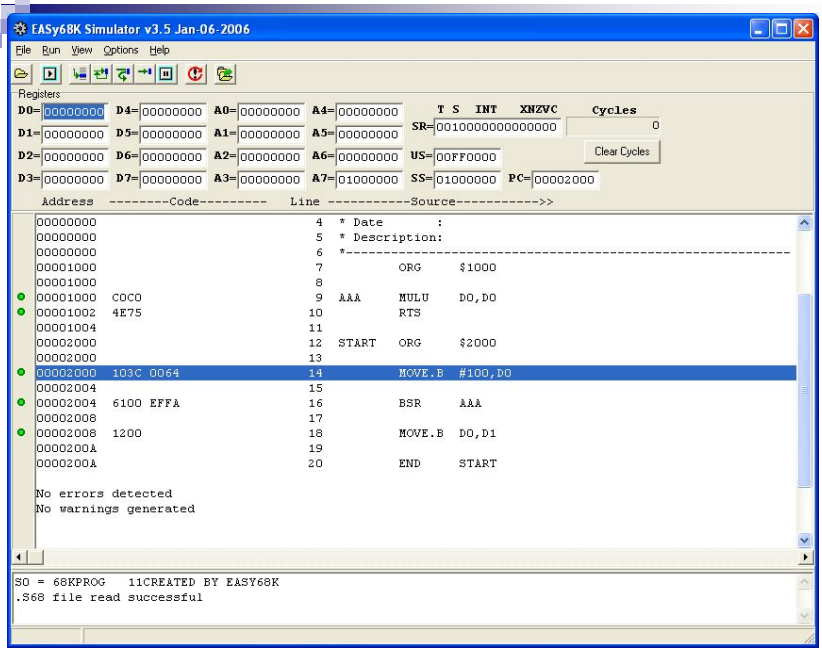
$FFFB	
$FFFD	
$FFFF	

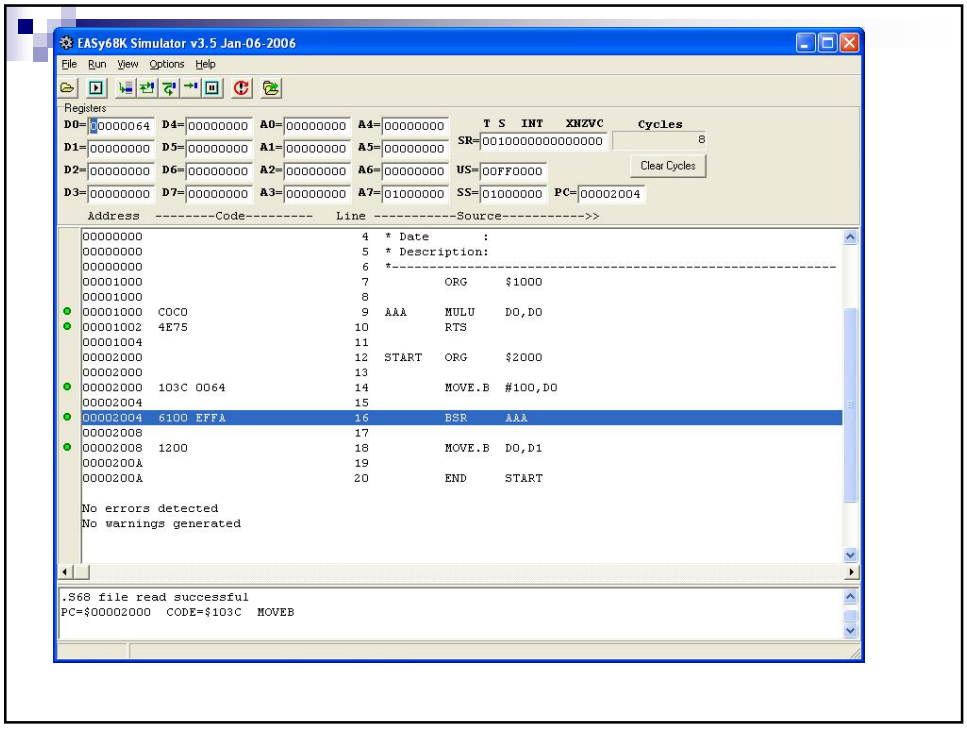
SUBROUTINE SUBR

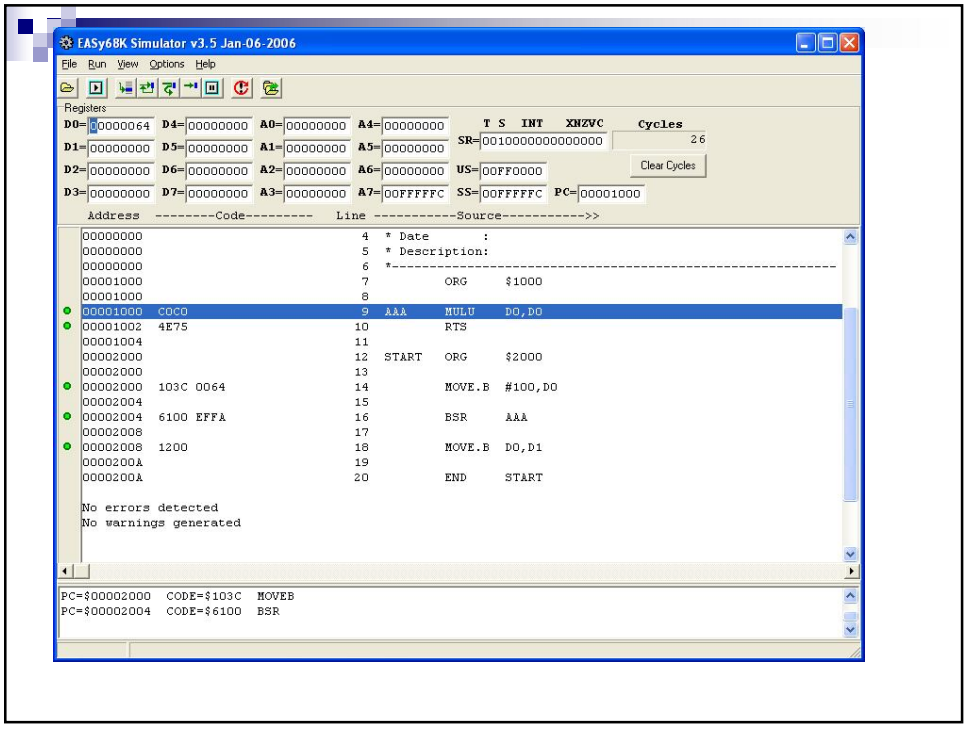
$2000	MULU D0,D0
$2002	RTS
$2004	

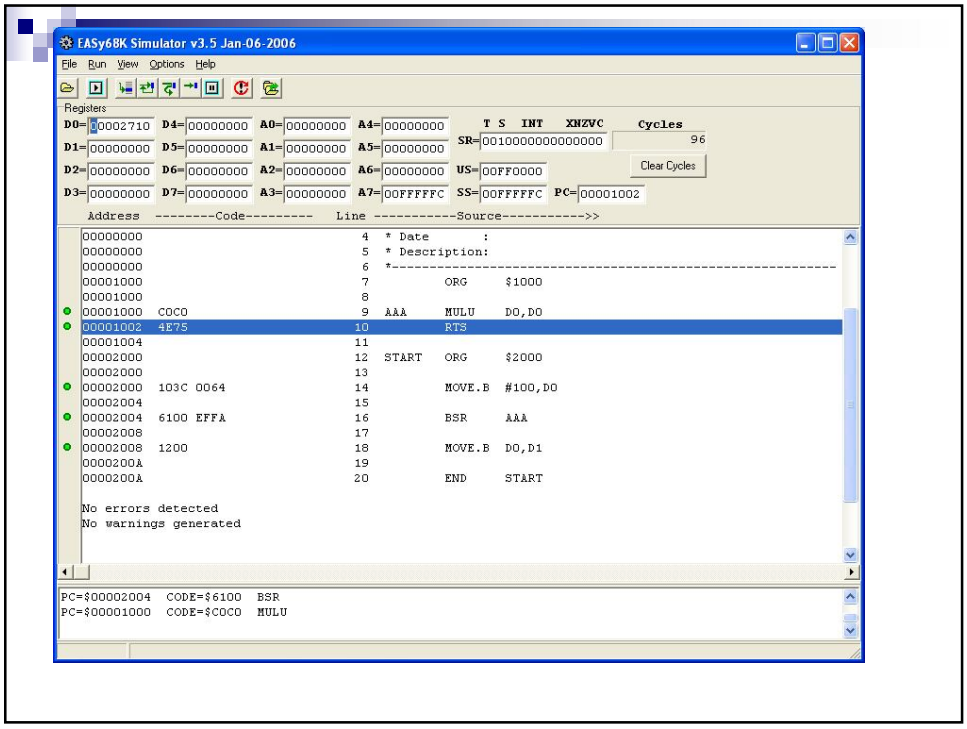
Sample Program

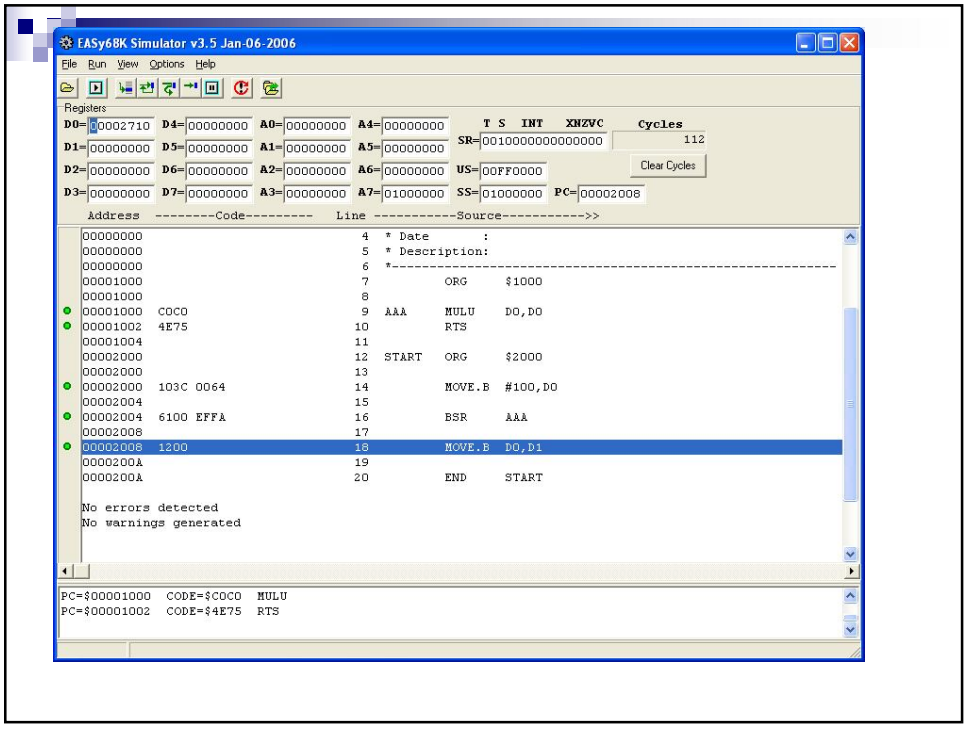
```
                ORG             $1000
AAA             MULU            D0,D0
                RTS

START           ORG             $2000
                MOVE.B          #100,D0
                BSR             AAA
                MOVE.B          D0,D1

                END             START
```




EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000064 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 8
US=00FF0000
SS=01000000 PC=00002004
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *----------
00001000 7 ORG $1000
00001000 8
00001000 COC0 9 AAA MULU D0,D0
00001002 4E75 10 RTS
00001004 11
00002000 12 START ORG $2000
00002000 13
00002000 103C 0064 14 MOVE.B #100,D0
00002004 15
00002004 6100 EFFA 16 BSR AAA
00002008 17
00002008 1200 18 MOVE.B D0,D1
0000200A 19
0000200A 20 END START
No errors detected
No warnings generated
.S68 file read successful
PC=$00002000 CODE=$103C MOVEB


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00000064 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=00FFFFFC
T S INT XNZVC Cycles
SR=0010000000000000 26
US=00FF0000
SS=00FFFFFC PC=00001000
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *----------
00001000 7 ORG $1000
00001000 8
00001000 COC0 9 AAA MULU D0,D0
00001002 4E75 10 RTS
00001004 11
00002000 12 START ORG $2000
00002000 13
00002000 103C 0064 14 MOVE.B #100,D0
00002004 15
00002004 6100 EFFA 16 BSR AAA
00002008 17
00002008 1200 18 MOVE.B D0,D1
0000200A 19
0000200A 20 END START
No errors detected
No warnings generated
PC=$00002000 CODE=$103C MOVEB
PC=$00002004 CODE=$6100 BSR


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00002710 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=00FFFFFC
T S INT XNZVC Cycles
SR=0010000000000000 96
US=00FF0000
SS=00FFFFFC PC=00001002
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *----------
00001000 7 ORG $1000
00001000 8
00001000 COC0 9 AAA MULU D0,D0
00001002 4E75 10 RTS
00001004 11
00002000 12 START ORG $2000
00002000 13
00002000 103C 0064 14 MOVE.B #100,D0
00002004 15
00002004 6100 EFFA 16 BSR AAA
00002008 17
00002008 1200 18 MOVE.B D0,D1
0000200A 19
0000200A 20 END START
No errors detected
No warnings generated
PC=$00002004 CODE=$6100 BSR
PC=$00001000 CODE=$C0C0 MULU


EASy68K Simulator v3.5 Jan-06-2006
File Run View Options Help
Registers
D0=00002710 D4=00000000 A0=00000000 A4=00000000
D1=00000000 D5=00000000 A1=00000000 A5=00000000
D2=00000000 D6=00000000 A2=00000000 A6=00000000
D3=00000000 D7=00000000 A3=00000000 A7=01000000
T S INT XNZVC Cycles
SR=0010000000000000 112
US=00FF0000
SS=01000000 PC=00002008
Clear Cycles
Address --------Code--------- Line ----------Source---------->>
00000000 4 * Date :
00000000 5 * Description:
00000000 6 *----------
00001000 7 ORG $1000
00001000 8
00001000 COC0 9 AAA MULU D0,D0
00001002 4E75 10 RTS
00001004 11
00002000 12 START ORG $2000
00002000 13
00002000 103C 0064 14 MOVE.B #100,D0
00002004 15
00002004 6100 EFFA 16 BSR AAA
00002008 17
00002008 1200 18 MOVE.B D0,D1
0000200A 19
0000200A 20 END START
No errors detected
No warnings generated
PC=$00001000 CODE=$C0C0 MULU
PC=$00001002 CODE=$4E75 RTS

Conclusion

Summary of Instructions

	BRA	JMP
Application	Perform unconditional branching.	
Method	Modifies PC to point to new location.	
Addressing Mode	Relative	Absolute
Limitations	Limited addressing range (16-bit signed displacement).	Unlimited

Summary of Instructions

	Bcc	DBcc	Scc
Description	Performs conditional branching	Same as Bcc, has counter.	Conditionally sets/clears lower byte based on test result.
Method	Branch to label if condition true, Else execute next instruction.	Decrement, branch to label if condition false, execute next if condition true or counter = -1.	Set byte to $FF is condition true, else set to $00.
Applications	If, if..else, for, while	For	Testing conditions.
Limitations	Limited addressing range (16-bit signed displacement).		N/A

Summary of Instructions

	BSR	JSR
Application	Branch unconditionally to subroutine.	
Branching Method	Modifies PC to point to new location. Saves return address to stack.	
Addressing Mode	Relative	Absolute
Limitations	Limited addressing range (16-bit signed displacement).	Unlimited

	RTS	RTR
Description	Used to return from subroutine.	
Method	Loads address from stack, put into PC.	Loads address & PC from stack, put into PC, CCR.

The End

Please read:

Antonakos, pg. 83-90.