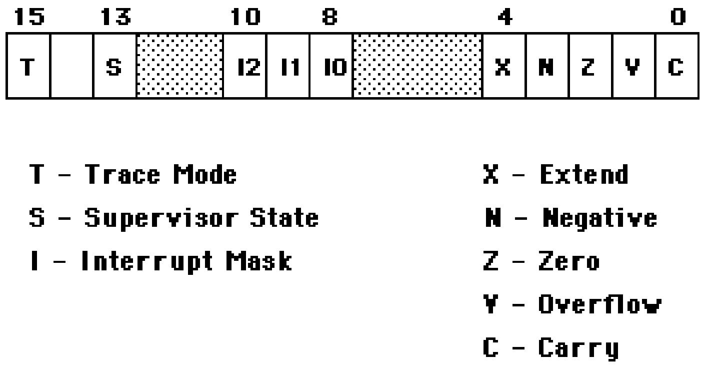
Lecture 5: Program control

- Condition Code Register (review)
- Program control instructions (review)
- Unconditional branch
- Simple branch instructions
- TST and CMP instructions
- Signed comparison branches
- Unsigned comparison branches
- Structured Programming
  - IF..THEN..ELSE
  - WHILE..DO
  - FOR loop

Status/Condition Code Register (review)

- **More significant byte: SR**
  - Only modifiable in supervisor mode
  - Details in later sections
- **Least significant byte: CCR**
  - For user-level programs
  - Behavior depends on instruction



Bit	Meaning
T	Tracing for run-time debugging
S	Supervisor or User Mode
I	System responds to interrupts with a level higher than I
C	Set if a carry or borrow is generated. Cleared otherwise
V	Set if a signed overflow occurs. Cleared otherwise
Z	Set if the result is zero. Cleared otherwise
N	Set if the result is negative. Cleared otherwise
X	Retains information from the carry bit for multi-precision arithmetic

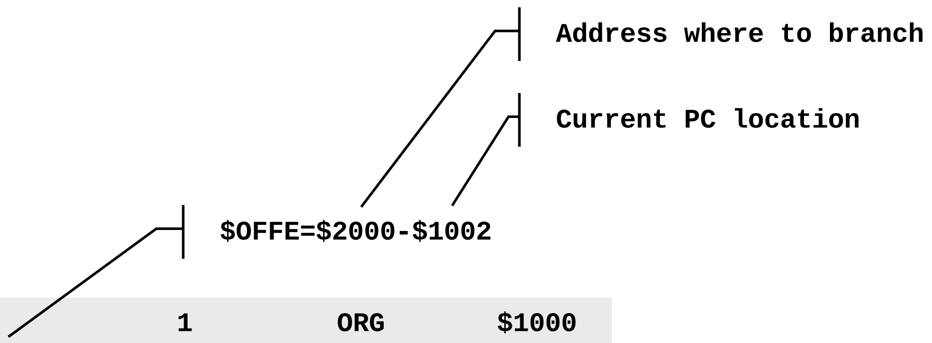
Program flow (review)

INSTR.	DESCRIPTION
BRA	BRA (branch always) implements an unconditional branch, relative to the PC. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, BRA **cannot** be used.
B*cc*	B*cc* (branch conditional) is used whenever program execution must follow one of two paths depending on a condition. The condition is specified by the mnemonic *cc*. The offset is expressed as an 8- or 16-bit signed integer. If the destination is outside of a 16-bit signed integer, B*cc* **cannot** be used.
BSR RTS	BSR branches to a subroutine. The PC is saved on the stack before loading the PC with the new value. RTS is use to return from the subroutine by restoring the PC from the stack.
JMP	JMP (jump) is similar to BRA. The only difference is that BRA uses only relative addressing, whereas JMP has more addressing modes, including absolute address (see reference manual).

cc	CONDITION	BRANCH TAKEN IF
CC	Carry clear	C=0
CS	Carry set	C=1
NE	Not equal	Z=0
EQ	Equal	Z=1
PL	Plus	N=0
MI	Minus	N=1
HI	Higher than	C*Z*=1
HS	Higher than or same as	C=0
LO	Lower	C=1
LS	Lower than or same as	C+Z=1
GT	Greater than	NVZ*+(NVZ)*=1
LT	Less than	NV*+N*V=1
GE	Greater than or equal to	NV*+N*V=0
LE	Less than or equal to	Z+(NV*+N*V)=1
VC	Overflow clear	V=0
VS	Overflow set	V=1
T	Always true	Always
F	Always false	Never

Unconditional branch (BRA)

- Program control is transferred unconditionally to the label specified in the BRA instruction
- Equivalent to the GOTO construct in high-level languages
- Example



```
00001000                  1          ORG       $1000
00001000   6000 0FFE      2          BRA       NEW_PC
00002000                  3          ORG       $2000
00002000   203C 00000000  4 NEW_PC   MOVE.L    #0,D0
00002006   4E75           5          RTS
00002008                  6          END
```

sim68k run of BRA example

```
Copyright (C) Livadas and Ward, 1992.  Author Wayne Wolf
Version 2.3
SIM68000  2.3 > LO L5.S
SIM68000  2.3 > .PC 1000
SIM68000  2.3 > DF

PC=00001000 SR=0000=..0..... US=00007000 SS=00007E00
D0=AE057F72 D1=1D2F73FC D2=752E7EDD D3=8565812C
D4=77ED68FA D5=6E97E64C D6=7FEB8022 D7=942C8974
A0=905D6857 A1=99AF78C7 A2=D741750B A3=88309416
A4=D8D09853 A5=6FB074A6 A6=5A96516A A7=00007000
----------------00001000 6000 0FFE                    BRA     $002000

SIM68000  2.3 > T
                                 PC loaded with new address
PC=00002000 SR=8000=T.0..... US=00007000 SS=00007E00
D0=AE057F72 D1=1D2F73FC D2=752E7EDD D3=8565812C
D4=77ED68FA D5=6E97E64C D6=7FEB8022 D7=942C8974
A0=905D6857 A1=99AF78C7 A2=D741750B A3=88309416
A4=D8D09853 A5=6FB074A6 A6=5A96516A A7=00007000
----------------00002000 203C 00000000            MOVE.L  #$00000000,D0

SIM68000  2.3 >

PC=00002006 SR=8004=T.0..Z.. US=00007000 SS=00007E00
D0=00000000 D1=1D2F73FC D2=752E7EDD D3=8565812C
D4=77ED68FA D5=6E97E64C D6=7FEB8022 D7=942C8974
A0=905D6857 A1=99AF78C7 A2=D741750B A3=88309416
A4=D8D09853 A5=6FB074A6 A6=5A96516A A7=00007000
----------------00002006 4E75                     RTS

SIM68000  2.3 >
```

Simple branch instructions (Bcc)

- **Branching is performed depending on individual value of Z,N,V or C**
- **If condition is true**
  - PC is loaded with label
  - Otherwise PC=PC+1
- **Example**

```
00001000                      1         ORG      $1000
00001000  203C 00000000       2         MOVE.L   #0,D0
00001006  6700 0004           3         BEQ      LABEL1
0000100A  4E71                4         NOP
0000100C  0480 00000001       5 LABEL1  SUB.L    #1,D0
00001012  6B00 0004           6         BMI      LABEL2
00001016  4E71                7         NOP
00001018  303C 7FFF           8 LABEL2  MOVE.W   #$7FFF,D0
0000101C  0640 7FFF           9         ADD.W    #$7FFF,D0
00001020  6900 0004          10         BVS      LABEL3
00001024  4E71               11         NOP
00001026  4E75               12 LABEL3  RTS
00001028                     13         END
```

cc	CONDITION	BRANCH TAKEN IF
CC	Carry clear	C=0
CS	Carry set	C=1
NE	Not equal	Z=0
EQ	Equal	Z=1
PL	Plus	N=0
MI	Minus	N=1
HI	Higher than	C*Z*=1
HS	Higher than or same as	C=0
LO	Lower	C=1
LS	Lower than or same as	C+Z=1
GT	Greater than	NVZ*+(NVZ)*=1
LT	Less than	NV*+N*V=1
GE	Greater than or equal to	NV*+N*V=0
LE	Less than or equal to	Z+(NV*+N*V)=1
VC	Overflow clear	V=0
VS	Overflow set	V=1
T	Always true	Always
F	Always false	Never

sim68k run of Bcc example

```
MC68000/ECB Simulator.
Copyright (C) Livadas and Ward, 1992.  Author Wayne Wolf
Version 2.3
SIM68000  2.3 > LO L5.S
SIM68000  2.3 > .PC 1000
SIM68000  2.3 > DF

PC=00001000 SR=0000=..0..... US=00007000 SS=00007E00
D0=8C41C60E D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001000 203C 00000000      MOVE.L  #$00000000,D0

SIM68000  2.3 > T

PC=00001006 SR=8004=T.0..Z.. US=00007000 SS=00007E00
D0=00000000 D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001006 6700 0004          BEQ     $00100C

SIM68000  2.3 >

PC=0000100C SR=8004=T.0..Z.. US=00007000 SS=00007E00
D0=00000000 D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------0000100C 0480 00000001      SUB.L   #$00000001,D0

SIM68000  2.3 >

PC=00001012 SR=8019=T.0XN..C US=00007000 SS=00007E00
D0=FFFFFFFF D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001012 6B00 0004          BMI     $001018
```

Last operation produced a zero result

```
PC=00001012 SR=8019=T.0XN..C US=00007000 SS=00007E00
D0=FFFFFFFF D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001012 6B00 0004          BMI     $001018

SIM68000  2.3 >

PC=00001018 SR=8019=T.0XN..C US=00007000 SS=00007E00
D0=FFFFFFFF D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001018 303C 7FFF          MOVE.W  #$7FFF,D0

SIM68000  2.3 >

PC=0000101C SR=8010=T.0X.... US=00007000 SS=00007E00
D0=FFFF7FFF D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------0000101C 0640 7FFF          ADD.W   #$7FFF,D0

SIM68000  2.3 >

PC=00001020 SR=800A=T.0.N.V. US=00007000 SS=00007E00
D0=FFFFFFFE D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001020 6900 0004          BVS     $001026

SIM68000  2.3 >

PC=00001026 SR=800A=T.0.N.V. US=00007000 SS=00007E00
D0=FFFFFFFE D1=64D39D61 D2=6EA09F13 D3=6F9E8C40
D4=5934BDB5 D5=6D4920FD D6=C3369559 D7=2CCFA8C4
A0=A1FABD23 A1=A7D1A23C A2=9DFE845A A3=D19176D3
A4=7191A733 A5=23FDCA40 A6=7ACD68F7 A7=00007000
----------------00001026 4E75               RTS

SIM68000  2.3 >
```

Last operation produced a negative result

Last operation produced overflow

Testing prior to branching

- **TST instruction**
  - format: **`TST.<size> <ea>`**
  - updates the value of CCR according to the operand
- **Why do we need TST?**
  - Branching decision depends on current CCR value
    - What if CCR is modified prior to BRA instruction?

```
ADD.W       #$6000,D0
JSR         MY_SUBROUTINE
BPL         PL_LABEL
```

    - Solution*

```
ADD.W       #$6000,D0
JSR         MY_SUBROUTINE
TST.W       DO
BPL         PL_LABEL
```

***This will not work if the value of D0 is modified in the subroutine!!!**

CMP instruction

- **Format**

```
CMP.<size>  <ea>,Dn
CMPI.<size> #N,<ea>
```

- **What does CMP do?**
  - CMP subtracts the contents of the source operand from the destination operand **without changing the destination**
  - after the subtraction, the CCR is updated

Signed comparison branches

- **Allows the programmer to use high-level relations such as**
  - LESS THAN
  - GREATER THAN
  - EQUAL
  - and so forth
- **CCR bits can be assigned with the CMP instruction**

cc	CONDITION	BRANCH TAKEN IF	?
CC	Carry clear	C=0	
CS	Carry set	C=1	
NE	Not equal	Z=0	
EQ	Equal	Z=1	
PL	Plus	N=0	
MI	Minus	N=1	
HI	Higher than	C*Z*=1	
HS	Higher than or same as	C=0	
LO	Lower	C=1	
LS	Lower than or same as	C+Z=1	
GT	Greater than	NVZ*+(NVZ)*=1	D>S
LT	Less than	NV*+N*V=1	D<S
GE	Greater than or equal to	NV*+N*V=0	D≥S
LE	Less than or equal to	Z+(NV*+N*V)=1	D≤S
VC	Overflow clear	V=0	
VS	Overflow set	V=1	
T	Always true	Always	
F	Always false	Never	

Unsigned comparison branches

- **Unsigned branches are needed to work with unsigned numbers, characters and addresses**
  - Unsigned numbers are always positive, so the N and V bits should not be tested
  - GT, LT, GE and LE check N and V
- **CCR bits can be assigned with the CMP instruction**

cc	**CONDITION**	**BRANCH TAKEN IF**	**?**
CC	Carry clear	C=0	
CS	Carry set	C=1	
NE	Not equal	Z=0	
EQ	Equal	Z=1	
PL	Plus	N=0	
MI	Minus	N=1	
HI	Higher than	C*Z*=1	D>S
HS	Higher than or same as	C=0	D≥S
LO	Lower	C=1	D<S
LS	Lower than or same as	C+Z=1	D≤S
GT	Greater than	NVZ*+(NVZ)*=1	
LT	Less than	NV*+N*V=1	
GE	Greater than or equal to	NV*+N*V=0	
LE	Less than or equal to	Z+(NV*+N*V)=1	
VC	Overflow clear	V=0	
VS	Overflow set	V=1	
T	Always true	Always	
F	Always false	Never	

Example of complex branches

```
00001000                    ORG      $1000
00001000  203C 00000064     MOVE.L   #100,D0     ;signed data
00001006  223C FFFFFF38     MOVE.L   #-200,D1    ;signed data
0000100C  B081              CMP.L    D1,D0
0000100E  6E00 0004         BGT      LABEL1      ;jump to LABEL1 if D0>D1
00001012  4E71              NOP
00001014  203C 00000064   LABEL1  MOVE.L   #100,D0     ;unsigned data
0000101A  223C 00000032     MOVE.L   #50,D1      ;unsigned data
00001020  B081              CMP.L    D1,D0
00001022  6D00 0004         BLT      LABEL2      ;jump to LABEL2 if D0<D1
00001026  4E71              NOP
00001028  4E75            LABEL2  RTS
0000102A                    END
```

sim68k run of complex branches

```
MC68000/ECB Simulator.
Copyright (C) Livadas and Ward, 1992.  Author Wayne Wolf
Version 2.3
SIM68000  2.3 > LO L5.S
SIM68000  2.3 > .PC 1000
SIM68000  2.3 > DF

PC=00001000 SR=0000=..0..... US=00007000 SS=00007E00
D0=6144A291 D1=C8A21F51 D2=657D79F8 D3=E8BA4017
...
----------------00001000 203C 00000064      MOVE.L  #$00000064,D0

SIM68000  2.3 > T

PC=00001006 SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=C8A21F51 D2=657D79F8 D3=E8BA4017
...
----------------00001006 223C FFFFFF38      MOVE.L  #$FFFFFF38,D1

SIM68000  2.3 >

PC=0000100C SR=8008=T.0.N... US=00007000 SS=00007E00
D0=00000064 D1=FFFFFF38 D2=657D79F8 D3=E8BA4017
...
----------------0000100C B081               CMP.L   D1,D0

SIM68000  2.3 >

PC=0000100E SR=8001=T.0....C US=00007000 SS=00007E00
D0=00000064 D1=FFFFFF38 D2=657D79F8 D3=E8BA4017
...
----------------0000100E 6E00 0004          BGT     $001014

SIM68000  2.3 >

PC=00001014 SR=8001=T.0....C US=00007000 SS=00007E00
D0=00000064 D1=FFFFFF38 D2=657D79F8 D3=E8BA4017
...
----------------00001014 203C 00000064      MOVE.L  #$00000064,D0
```

```
SIM68000  2.3 >

PC=0000101A SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=FFFFFF38 D2=657D79F8 D3=E8BA4017
...
----------------0000101A 223C 00000032      MOVE.L  #$00000032,D1

SIM68000  2.3 >

PC=00001020 SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=00000032 D2=657D79F8 D3=E8BA4017
...
----------------00001020 B081               CMP.L   D1,D0

SIM68000  2.3 >

PC=00001022 SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=00000032 D2=657D79F8 D3=E8BA4017
...
----------------00001022 6D00 0004          BLT     $001028

SIM68000  2.3 >

PC=00001026 SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=00000032 D2=657D79F8 D3=E8BA4017
...
----------------00001026 4E71               NOP

SIM68000  2.3 >

PC=00001028 SR=8000=T.0..... US=00007000 SS=00007E00
D0=00000064 D1=00000032 D2=657D79F8 D3=E8BA4017
...
----------------00001028 4E75               RTS
```

Decrement and branch instruction

- **Syntax:** `DBcc Dn,<LABEL>` **(word operation!!!)**
- **Action:**

```
if (cc==TRUE) {
  PC=PC+1;
} else {
    Dn.W = Dn.W - 1;
    if (Dn.W == -1) {
      PC=PC+1;
    }
    else {
      PC=LABEL;
    }
}
```

- **Example**

```
                      MOVE.W      #100,D0                ;load D0 with 100
NEXT                  <loop_instructions>                ;execute loop instructions
                      DBF         D0,NEXT                ;decr. D0 and branch if not -1
```

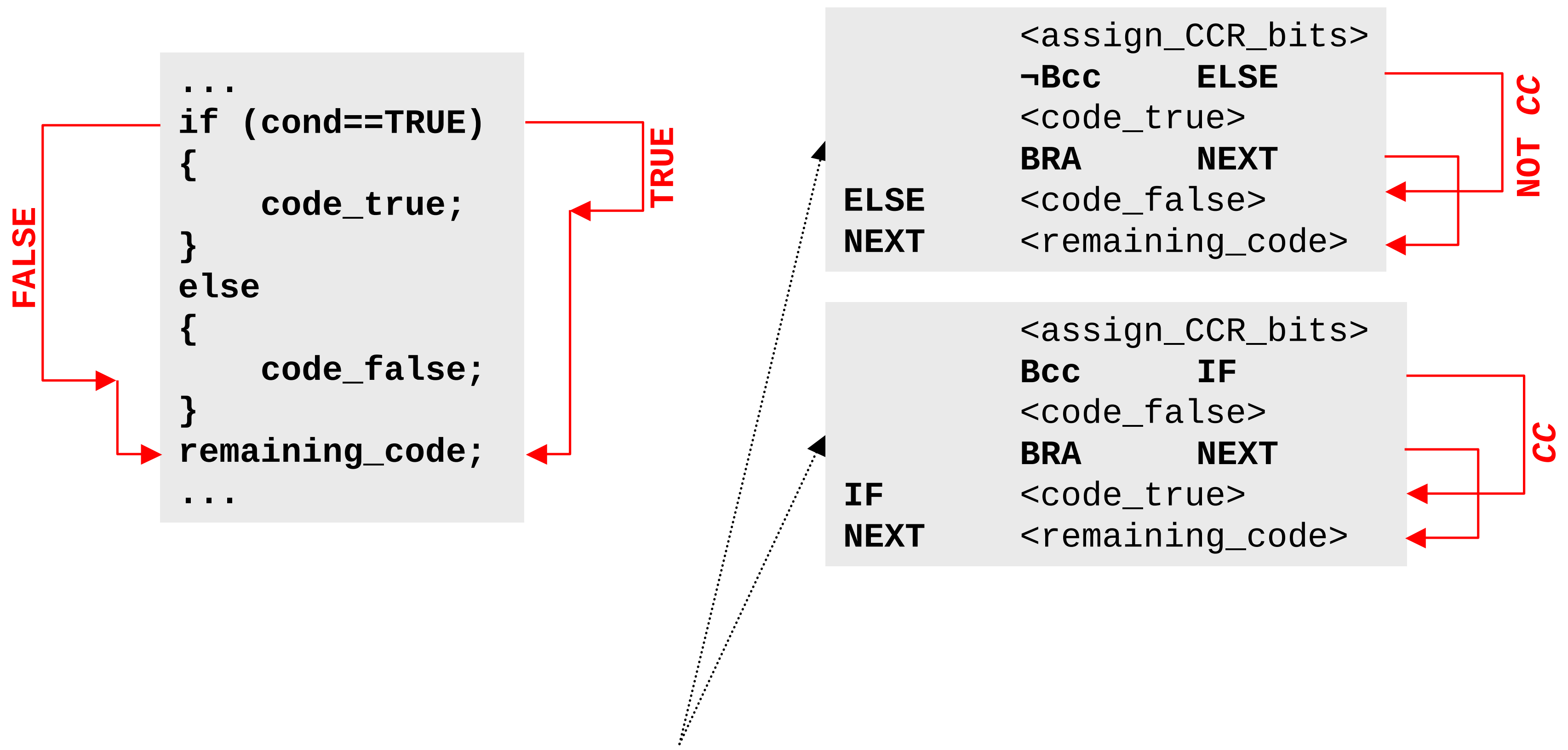
DBF ALWAYS causes the branch to be taken until the contents of D0 are -1

Structured programming

- **Conditional execution**
  - Most computer algorithms contain decision points where code is conditionally executed depending on the status of program variables
  - Conditional execution constructs are available in high-level languages
    - IF..THEN..ELSE
    - WHILE..DO
    - FOR loop
- **In the MC68000 conditional execution is implemented through**
  - Condition Code Register (CCR) and
  - Branching instructions

IF..THEN..ELSE

High-Level Language Construct | **Assembly Language Construct**



```
...
if (cond==TRUE)
{
     code_true;
}
else
{
     code_false;
}
remaining_code;
...
```

```
             <assign_CCR_bits>
             ¬Bcc      ELSE
             <code_true>
             BRA       NEXT
ELSE         <code_false>
NEXT         <remaining_code>
```

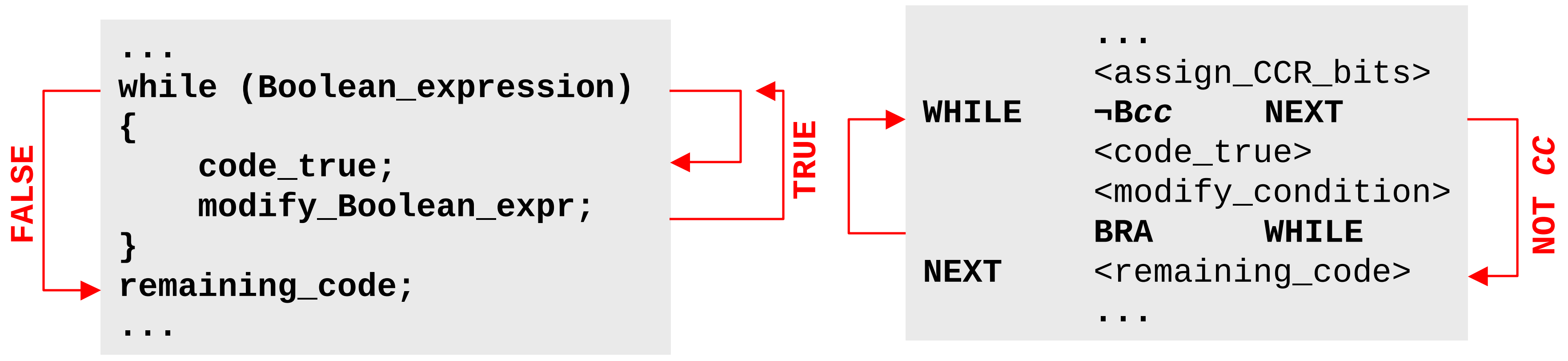
```
             <assign_CCR_bits>
             Bcc       IF
             <code_false>
             BRA       NEXT
IF           <code_true>
NEXT         <remaining_code>
```

Two different implementations

WHILE..DO

High-Level Language Construct

Assembly Language Construct

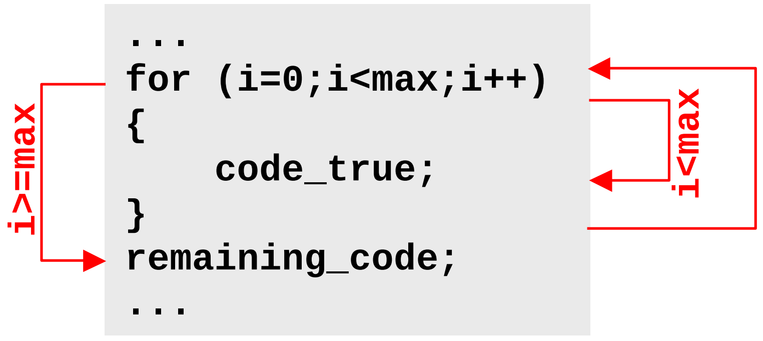


```
...
while (Boolean_expression)
{
    code_true;
    modify_Boolean_expr;
}
remaining_code;
...
```

```
             ...
             <assign_CCR_bits>
WHILE        ¬Bcc      NEXT
             <code_true>
             <modify_condition>
             BRA       WHILE
NEXT         <remaining_code>
             ...
```

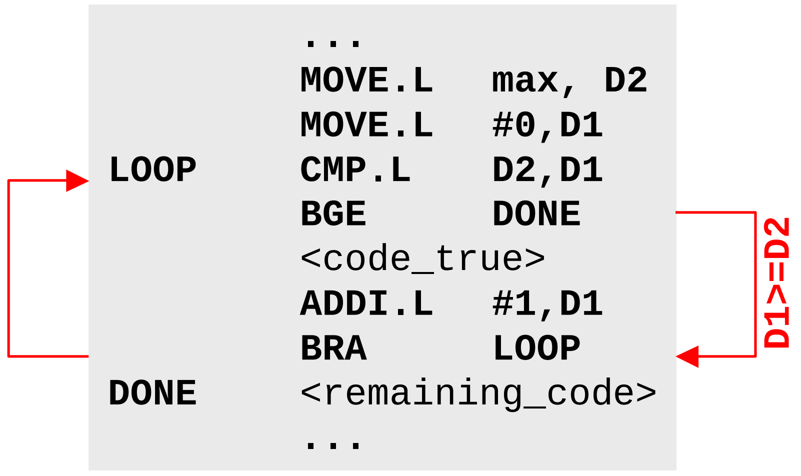
FOR loop

High-Level Language Construct



```
...
for (i=0;i<max;i++)
{
    code_true;
}
remaining_code;
...
```

Assembly Language Construct



```
         ...
         MOVE.L   max, D2
         MOVE.L   #0,D1
LOOP     CMP.L    D2,D1
         BGE      DONE
         <code_true>
         ADDI.L   #1,D1
         BRA      LOOP
DONE     <remaining_code>
         ...
```

Example

- **Calculate $\sum_{n=1}^{10} n^2$ and store it in D0**
- **Solution**

```
        ORG     $1000
        CLR.L   D0
        MOVE.L  #1,D1     ;D1 used as a counter (n)
LOOP    CMP.L   MAX,D1
        BHI     DONE      ;terminate when D1>max
        MOVE.L  D1,D2     ;make a copy of D1
        MULU.W  D2,D2     ;calculate D2^2
        ADD.L   D2,D0     ;D0=D0+D2^2
        ADDQ    #1,D1     ;increment counter
        BRA     LOOP      ;loop again
DONE    RTS
MAX     DC.L    10
        END
```