Branching Instruction

There are two types of Branch – unconditional and conditional branch

OP (4-bit)	Cond (4-bit)	Displacement (8-bit or 16-bit)

4-bit Cond (condition) for branching represents:

- Unconditional: 0000
- Conditional:
  - Branch on Equal: 0111
  - Branch on Greater Than: 1110
  - Branch on Less Than: 1101

Displacement: Short or Long displacement

If 8-bit displacement is $\neq 0$, it represents a short branch

0110	Cond	Displacement (8-bit)

If 8-bit displacement is $= 0$, it represents a long branch; the next word represents the displ.

0110	Cond	0000 0000	Displacement (16-bit)

Branch Instructions

Unconditional: bra label

Example: bra $4000 (default for word displacement)
bra.w $4000 (.w extension specify word displacement)
bra.b $4000 (.b extension specify byte displacement)
bra loop

Target PC = Current PC + 2 + displacement

Conditional: checks the condition code (C, V, Z, N, X) bits in Status Register

Branch on Equal: beq label

If $Z = 1$,
then PC ← PC + 2 + displacement
else PC ← PC + 2

Branch on Greater Than: bgt label

If $Z = 0$ and $N = V$,
then PC ← PC + 2 + displacement
else PC ← PC + 2

Branch on Less Than: blt label

If $N \neq V$,
then PC ← PC + 2 + displacement
else PC ← PC + 2

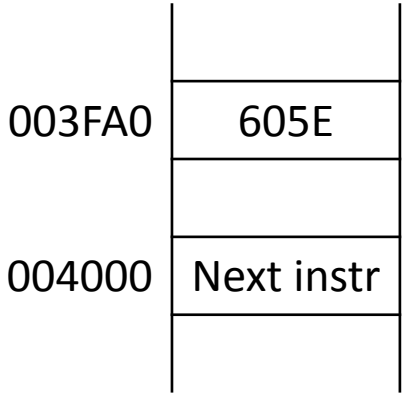
Calculating Target PC value

Target PC ← current PC + 2 + displacement

There is (+ 2) above since PC changes to PC + 2 after the first word of branch instruction has been accessed and decoded, irrespective of short or long branch

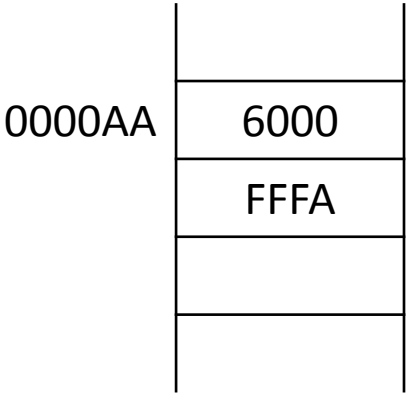
Example Calculation:

$ 0000 3FA0	Current PC
+ 0000 005E	8-bit displ
+ 2	
= 0000 4000	Target PC



Displacement can be forward (+ve) or backward (-ve)

$ 0000 00AA	Current PC
+ FFFF FFFA	16-bit sign-ext displ
+ 2	
= 0000 00A6	Target PC



Calculating displacement value

Target PC ← current PC + 2 + displacement

If the target PC value is known, which is the Label value, the displacement can be easily obtained from the above equation

Example: If instruction is **bra $4000** and current **PC = $003F40**,

then

$004000 = $003FA0 + displacement + 2

displacement = $004000 - $003FA0 – 2

= $5E

For short branch, the machine code is: 605E

For long branch, the machine code is: 6000 005E

Branch Example - 1

Given *x* and *y* two 16-bit values stored in registers *d1* and *d2*, respectively.

Write an assembly language segment to implement:

if *x = y* then *x := x + y*

else *y := x + y*

The assembly language part which implements the above is as follows:

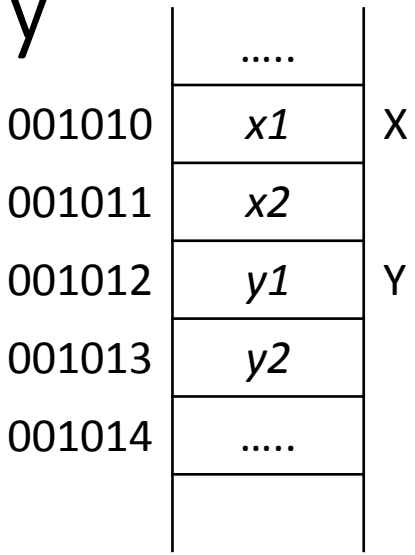
```
        ----
        cmp d1, d2      ; x = y ?
        beq then        ; go to then
else    add d1, d2      ; else y = x + y
        bra done        ; jump over
then    add d2, d1      ; then x = x + y
Done    ----
        ---
```

Branch Example – 2 with Hand Assembly

Given *x* and *y* two 16-bit values stored in memory. Implement:

if *x = y* then *x := x + y*

else *y := x + y*



PC			main	org $0000	; program starts at $0000
000000	9040			sub d0, d0	; d0 ← 0
000002	3040			movea d0, a0	; a0 ← 0
000004	3228	1010		move X(a0), d1	; d1 ← M[X + a0] = x
000008	3428	1012		move Y(a0), d2	; d2 ← M[Y + a0] = y
00000C	8441			cmp d1, d2	; x = y ?
00000E	6708			beq.b THEN	
000010	D442		ELSE	add d1, d2	; d2 = x + y
000012	3142	1012		move d2, Y(a0)	; M[Y + a0] ← d2
000016	6006			bra.b DONE	
000018	D241		THEN	add d2, d1	; d1 = x + y
00001A	3141	1010		move d1, X(a0)	; M[X + a0] ← d1
00001E	3028	1014	DONE	move Y+2(a0), d0	; d0 ← 3
000022	4E40			trap #0	; STOP

PC			org $1010	
001010	0064	X	dc 100	; defines x
001012	0062	Y	dc $62	; defines y
001014	0003		dc 3	; to stop
			end	

Another Branch Example

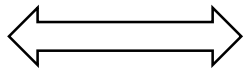
Given *x* and *y* two 16-bit values stored in memory X and Y respectively. Implement:

if *x* < *y* then *x* := *y* - *x*

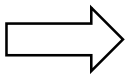
else *y* := *y* – *x*

```
        org $1000
        sub d1, d1
        movea d1, a1
        move X(a1), d0    ; d0 ← M[X]
        move Y(a1), d3    ; d3 ← M[Y]
        cmp d0, d3
        bgt THEN          ; y > x
ELSE    sub d0, d3        ; y – x
        move d3, Y(a1)    ; M[Y] ← y – x
        bra DONE
THEN    sub d0, d3        ; y – x
        move d3, d0
        move d0, X(a1)
DONE    …

  X     ds  1
  Y     ds  1
        end
```

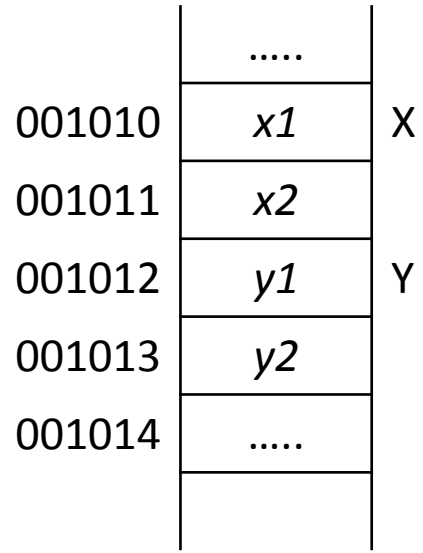


```
        cmp d0, d3
        blt ELSE
        beq ELSE
THEN    sub d0, d3
        move d3, d0
        move d0, X(a1)
        bra DONE
ELSE    sub d0, d3
        move d3, Y(a1)
DONE    ….
```



Shorter Alternative:

```
        sub d0, d3
        bgt THEN
ELSE    move d3, Y(a1)
        bra DONE
THEN    move d3, X(a1)
DONE    ….
```



Add	1101	dDn_3	001000	sDn_3
	1101	dAn_3	011000	sDn_3
Subtract	1001	dDn_3	001000	sDn_3
Multiply	1100	dDn_3	111000	sDn_3
Divide	1000	dDn_3	111000	sDn_3
Compare	1011	dDn_3	001000	sDn_3
Swap	0100100001000			dDn_3
Move (register to register)	0011	dDn_3	000000	sDn_3
	0011	dAn_3	001000	sDn_3
	0011	dDn_3	000001	sAn_3
	0011	dAn_3	001001	sAn_3
Move (register to memory)	0011	An_3	101001	sAn_3
	$Displacement_{16}$			
	0011	An_3	101000	sDn_3
	$Displacement_{16}$			
Move (memory to register)	0011	dAn_3	001101	sAn_3
	$Displacement_{16}$			
	0011	dDn_3	000101	An_3
	$Displacement_{16}$			
Move (memory to memory)	0011	An_3	101101	An_3
	$s\text{-}Displacement_{16}$			
	$d\text{-}Displacement_{16}$			
Branch (unconditional)	01100000	$Displacement_8$		
Branch on equal	01100111	$Displacement_8$		
Branch on greater	01101110	$Displacement_8$		
Branch on less	01101101	$Displacement_8$		
Stop, dump	010011100100			0000