6 Branch Instructions

The branch instructions AB, ABL, ABR, ABI, RB and RF branch to an address or the contents of an address or register when a certain condition is fulfilled. If that condition does not arise the program determines the next instruction to be executed.
The condition is given by a number from 1 through 7 or by one or two letters.
The following table gives a survey:

Condition Notation

Cond. reg. contents	(cnd)			
	GENERAL	ARITHM.	COMPARE	I/O
0	(0)	(Z) Zero	(E) Equal	(A) Accepted
1	(1)	(P) Pos.	(G) Greater	(R) Refused
2	(2)	(N) Neg.	(L) Less	–
3	(3)	(O) Overfl.	–	(U) Unknown
	NOT – Condition			
≠0	(4)	(NZ) Not Zero	(NE) Not Equal	(NA) Not Accepted
≠1	(5)	(NP) Not Pos.	(NG) Not Greater	(NR) Not Refused
≠2	(6)	(NN) Not Neg.	(NL) Not Less	–
n.s.	(7)	Unconditional		

Note:
The instruction counter P always points to the next instruction to be executed. Wherever in the description the notation (P) + 2 (or 4) appears, the hardware function is meant.
When the following program must be assembled calculate the displacement in locations as follows:

```
BEGIN   EQU     *
        HLT
        LDK     A1,/000A
        SUK     A1,2
        RF(Z)   *+4
        RB      *-4
        ABL     *-8
        END     START
```

where *+4 refers to ABL
*–4 refers to SUK
*–8 refers to LDK

When the same program is to be put in memory with the toggle switches the value for RF(Z) *+4 is 5002 and not 5004 as the P-register is already pointing to the next instruction.
The value for RB *–4 must be 5F06 and not 5F04, as the P-register is already pointing to the next instruction.
The address in the ABL instruction must be the relative address pointing to LDK.

The values put in memory for the program listed above must be:

```
207F    START   HLT
010A            LDK     A1,/000A
1902            SUK     A1,2
5002            RF(Z)   *+4
5F06            RB      *-4
8F20            ABL     *-8
0002
                END     START
```

AB
ABL

Absolute conditional branch

AB
ABL

P851M
P852M
P856M
P857M

Syntax:	[label]␣ AB [(cnd)]␣ k	– T8
	[label]␣ ABL [(cnd)]␣ Ik	– T2

This instruction means that the next instruction to be executed is found either at the address specified by the constant (k indicating one of the first 256 addresses of the memory, Ik being specified in the word following the instruction) or in normal sequence, depending on the (cnd) and the contents of the condition register.
If (cnd) is equal to (7) the next instruction is at the effective memory address. Note that if (cnd) is omitted, the default value is (7).
The least significant bit in either constant, is always zero (word addressing).
See also table and note on page 6.0.1.

Effective branch:	*Type*	*Function*
	T8	k → P
	T2	Ik → P

No branch:	*Type*	*Function*
	T8	(P) + 2 → P
	T2	(P) + 4 → P

Condition register: Unchanged

bit	0	1			4	5 7	8 14	15
T8	0	0	0	0	1	CND	k	0

bit	0	1			4	5 7	8	9	10	11			14	15
T2	1	0	0	0	1	CND	▨	0	1	0	0	0	0	0

ABR — *Absolute conditional branch to register* — ABR

P851M
P852M
P856M
P857M

Syntax: [label]␣ABR [(cnd)] [*]␣r2

This instruction indicates that the address of next instruction to be executed is found either in the register specified by r2 or at the memory address indicated by the register or in normal sequence depending on (cnd) and the contents of the condition register.
If (cnd) = (7), the next instruction is at the effective memory address (unconditional branch).
If (cnd) is omitted, the defautl value is (7).
See also table and note on page 6.0.1.

Effective branch:

Type	*Function*	*MD*	*I/s*	*Syntax*
T1	(r2) → P	00	n.s.	ABR(cnd) r2
T3	((r2)) → P	01	0	ABR(cnd)* r2

No branch:

Type	*Function*	*MD*	*I/s*
T1	(P) + 2 → P	00	n.s.
T3	(P) + 2 → P	01	0

Condition register: Unchanged

bit	0	1 … 4	5 … 7	8	9 … 10	11 … 14	15
	1	0 0 0 1	CND	▨	MD	r2	0

ABI — *Absolute branch indirect* — ABI

P851M
P852M
P856M
P857M

Syntax: [label]␣ABI [(cnd)] [*]␣m[, r2]

The address of the next instruction to be executed is found either at the effective memory address or in the next instruction, depending on (cnd) and the contents of the condition register.
If (cnd) = (7) (see below) the instruction branched to is always at the effective memory address (unconditional branch).
In all other cases the program must first fulfil a condition before the branch takes place. If (cnd) is omitted, the default value is (7).
See also table and note on page 6.0.1.

Effective branch:

Type	*Function*	*MD*	*Syntax*
T4	(m) → P	10	ABI(cnd) m
T5	(m + (r2)) → P	10	ABI(cnd) m, r2
T6	((m)) → P	11	ABI(cnd)* m
T7	((m + (r2))) → P	11	ABI(cnd)* m, r2

No branch:

Type	*Function*	*MD*
T4	(P) + 4 → P	10
T5	(P) + 4 → P	10
T6	(P) + 4 → P	11
T7	(P) + 4 → P	11

Condition register: Unchanged

bit	0	1 … 4	5 … 7	8	9 … 10	11 … 14	15
	1	0 0 0 1	CND	▨	MD	r2	0

RF

Relative forward conditional branch

RF

P851M
P852M
P856M
P857M

Syntax: [label]␣ RF [(cnd)]␣ m

This instruction indicates that the next instruction to be executed is found either at the effective memory address or in normal sequence, depending on (cnd) and the contents of the condition register. If (cnd) = (7) the next instruction can be found at the effective memory address (unconditional relative branch).
If (cnd) is omitted, the default value of (7) is assumed.

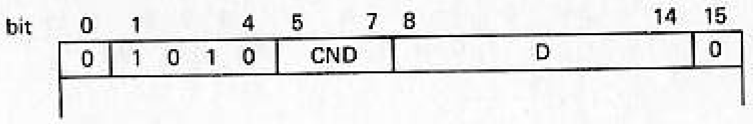
The assembler calculates from the effective memory address, a displacement D relative (forwards) to the current value of the instruction counter (P). This value is stored in bits 8–15 of the instruction as a positive number. Thus its maximum is 255. In programming terms, this means that this instruction can only be used to branch by ≤ 128 words.

See also table and note on page 6.0.1.

Type	*Function*	
T8	(P) + 2 + D → P	(branch effective)
T8	(P) + 2 → P	(no branch)

Example:
```
RF(Z)  END
RF(3)  *+12
```

Condition register: Unchanged



RB

Relative backwards conditional branch

RB

P851M
P852M
P856M
P857M

Syntax: [label]␣ RB [(cnd)]␣ m

This instruction means that the next instruction to be executed is found either at the effective memory address or in normal sequence, depending on (cnd) and the contents of the condition register. If (cnd) = (7) the next instruction to be executed is found at the effective memory address.
If (cnd) is omitted, the defaut value of (7) is assumed.

The assembler calculates from the effective memory address, a displacement D relative (backwards) to the current value of the instruction counter (P). This value is stored in bits 8–15 of the instruction as a positive number. Thus its maximum is 255. In programming terms, this means that this instruction can only be used to branch backwards by ≤ 128 words.

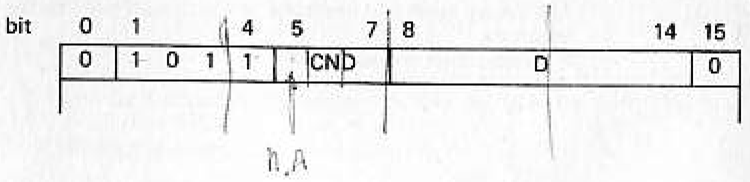
It should be noted that
␣ RB (cnd) ␣*
is equivalent to branch to itself and causes a continuous loop.

See also table and note on page 6.0.1.

Type	*Function*	
T8	(P) + 2 – D → P	(branch effective)
T8	(P) + 2 → P	(no branch)

Example:
```
RB(4)   LABEL
RB(NE)  *–2
```

Condition register: Unchanged



CF — *Call function* — CF

P851M
P852M
P856M
P857M

Syntax: [label]␣ CF ␣ r1, lk

This instruction provides a link to a subroutine by storing successively the contents of the P-register and the program status word (PSW) in a memory stack. The PSW contains, amongst other things, the priority level and condition register. The stack pointer is held in the register specified by r1 and is automatically updated. Then a branch is made to the address specified by lk.
The subroutine must be terminated by an RTN instruction to branch back to the main program.

Type	*Function*
T2	(P) → (r1),(r1) − 2 → r1
	(PSW) → (r1),(r1) − 2 → r1
	lk → P

Condition register: Unchanged. Its contents shows the result of a previous operation and is stored in the memory stack for use on return from the subroutine.

bit	0	1 — 4	5 — 8	9	10	11 — 14	15
	1	1 1 1 0	r1	0	1	0 0 0 0	1

Remark:
An interrupt 'stack overflow' is generated when r1 = A15 and the word address reached by the pointer = </100. Bit 13 is set in PSW.

* r1 must be ≠ 0.
* Restricted to system mode if r1 = A15.
* The system stack and user stack are both built towards the lower addresses.
  P is stored first and next PSW.

CFR — *Call function register* — CFR

P851M
P852M
P856M
P857M

Syntax: [label]␣ CFR[*]␣ r1, r2

This instruction provides a link to a subroutine by storing successively the contents of the P-register, which points to the next instruction of the main program, and the contents of the program status word (PSW) in a memory stack. The PSW contains, amongst other things the priority level and the condition register. The stack pointer held in the register specified by r1 is automatically updated by decreasing the stack pointer by 2, as the stack pointer is filled from the higher address towards the lower address.
Next a branch is made to the effective memory address specified by the contents of a register specified by r2.
The subroutine must be terminated by an RTN instruction to branch back to the main program.

Type	*Function*	*MD*	*Syntax*
T1,T3	(P) → (r1),(r1) − 2 → r1		
	(PSW) → (r1),(r1) − 2 → r1		
then:			
T1	(r2) → P	00	CFR r1, r2
T3	((r2)) → P	01	CFR* r1, r2

Condition register: Unchanged. Its contents shows the result of a previous operation and is stored in the memory stack for use on return from the subroutine.

bit	0	1 — 4	5 — 8	9 — 10	11 — 14	15
	1	1 1 1 0	r1	MD	r2	1

Remark:

* An interrupt 'stack overflow' is generated when r1 = A15 and the word address reached by the pointer = </100. Bit 13 in the PSW is set to 1.
* r1 must be ≠ 0.
* Restricted to system mode if r1 = A15.

CFI — *Call function indirect* — CFI

P851M
P852M
P856M
P857M

Syntax: [label]␣ CFI [*]␣ r1, m[, r2]

The instruction provides a link to a subroutine by storing successively the contents of the P-register, which points to the next instruction of the main program, and the contents of the program status word (PSW) in a memory stack. The PSW contains, amongst other things, the priority level and condition register. The stack pointer held in the register specified by r1 is automatically updated by decreasing the stack pointer by 2, as the stack pointer is filled from the higher address towards the lower address.
Next a branch is made to the contents of the effective memory address, i.e. the subroutine which has to be executed.
The subroutine must be terminated by an RTN instruction to branch back to the main program.

Type	*Function*		*MD*	*Syntax*
T4, T5	(P) → (r1)	(r1) − 2 → r1	n.a.	
T6, T7	(PSW) → (r1)	(r1) − 2 → r1	n.a.	

then:

T4	(m) → P	10	CFI r1, m
T5	(m + (r2)) → P	10	CFI r1, m, r2
T6	((m)) → P	11	CFI* r1, m
T7	((m + (r2))) → P	11	CFI* r1, m, r2

Condition register: Unchanged. Its contents shows the result of a previous operation and is stored in the memory stack for use on return from subprogram.

bit	0	1 … 4	5 … 8	9 … 10	11 … 14	15
	1	1 1 1 0	r1	MD	r2	1

Remark:
* An interrupt 'stack overflow' is generated when r1 = A15 and the word address reached by the pointer = </100. Bit 13 of the PSW is set to 1.
* r1 must be ≠ 0.
* Restricted to system mode if r1 = A15.

RTN — *Return from function (A1 . . . A14)* — RTN

P851M
P852M
P856M
P857M

Syntax: [label]␣ RTN ␣ r2

This instruction allows the return from a subroutine to the main program. It must be the last instruction of such a routine. The instruction reloads the P-register and CR-register which have previously been loaded into a memory stack by a Call Function instruction.

Type	*Function*	
T3	(r2) + 2	→ r2
	$((r2))_{0-5}$	→ PLR
	$((r2))_{6-7}$	→ CR
	(r2) + 2	→ r2
	((r2))	→ P

Condition register: Reloaded from stack, bits 6 and 7 of the PSW → CR.

bit	0	1 … 4	5 … 8	9	10	11 … 14	15
	1	1 1 1 0	0 0 0 0	0	1	r2	0

Remark:
r2 must be ≠ 0.

RTN *Return from function (A15)* RTN

P851M
P852M
P856M
P857M

Syntax: [label]␣ RTN ␣ A15

This instruction allows the return from an interrupt routine, a trap routine or subroutine. It must therefore be the last instruction of that routine. The instruction reloads the PSW and P-register which have previously been loaded into a memory stack by a Call Function instruction. The stack-pointer A15 is automatically updated.
On the P852M bit 9 of the PSW (ENB) is always set to 1. On the other computers bit 9 must be set, if required.
Note: By forcing bit 15 of the PSW to 1, the user may switch the machine from system mode to user mode (P851M, P856M and P857M).

Type	*Function (P852M)*		*Function (other)*	
T3	(A15) + 2	→ A15	(A15) + 2	→ A15
	((A15)) 0–5	→ PLR	((A15)) 0–5	→ PLR
	((A15)) 6,7	→ CR	((A15)) 6,7	→ CR
	bit 9 is set to 1	→ ENB	((A15)) 9	→ ENB
	SU bit does not exist		((A15)) 15	→ SU
	(A15) + 2	→ A15	(A15)+ 2	→ A15
	((A15))	→ P	((A15))	→ P

bit	0	1 4	5 8	9	10	11 14	15
	1	1 1 1 0	0 0 0 0	0	1	1 1 1 1	0

Remark:
r2 must be ≠ 0.

EXR *Execute register* EXR

P851M
P852M
P856M
P857M

Syntax: [label]␣ EXR[*]␣ r2

This instruction executes the instruction in r2 (T1) or pointed to by the contents of r2 (T3). r2 may not contain a double word instruction, a CF instruction, RTN instruction or another EXR, EX or EXK instruction.

Type	*Function*	*MD*	*Syntax*	
T1	(r2) is executed	00	EXR	r2
T3	((r2)) is executed	01	EXR*	r2

Condition register: CR is set by the instruction in (r2).

bit	0	1 4	5 8	9 10	11 14	15
	1	1 1 1 0	0 0 0 0	MD	r2	1

EXK

Execute constant

EXK

P851M
P852M
P856M
P857M

Syntax: [label]⌴ EXK ⌴ Ik

This instruction performs the operand instruction contained in Ik.
The memory address may not contain a double word instruction, a CF instruction, RTN instruction or another EXK, EX or EXR instruction.

Type	*Function*
T2	Ik is executed

Condition register: CR is set by the instruction in Ik.

bit	0	1 4	5 8	9 10	11 14	15
	1	1 1 1 0	0 0 0 0	0 1	0 0 0 0	1

EX

Execute

EX

P851M
P852M
P856M
P857M

Syntax: [label]⌴ EX[*]⌴ m[, r2]

This instruction executes the operand instruction contained in the effective memory address. The memory address may not contain a double word instruction, a CF instruction, RTN instruction or another EX, EXK, or EXR instruction.

Type	*Function*		*MD*	*Syntax*	
T4	(m)	is executed	10	EX	m
T5	(m + (r2))	"	10	EX	m, r2
T6	((m))	"	11	EX*	m
T7	((m + (r2)))	"	11	EX*	m, r2

Condition register: CR is set by the instruction in the effective memory address.

bit	0	1 4	5 8	9 10	11 14	15
	1	1 1 1 0	0 0 0 0	MD	r2	1