ADK
ADKL

Add constant

ADK
ADKL

P851M
P852M
P856M
P857M

Syntax:

[label] ␣ ADK ␣ r3, k — T8
[label] ␣ ADKL ␣ r1, lk — T2

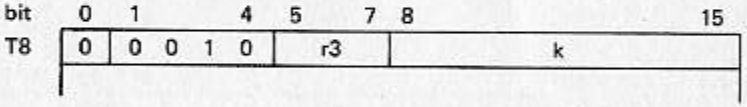
T8 The positive constant k is added to the contents of the register specified in r3. The result of the addition is placed in r3.

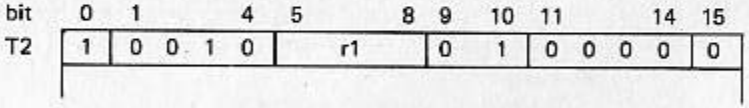
T2 The positive or negative constant lk is added to the contents of the register specified in r1. The result of the addition is placed in r1.

Type	*Function*	*Syntax*
T8	(r3) + k → r3	ADK r3, k
T2	(r1) + lk → r1	ADKL r1, lk

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow





Remark:
Restricted to system mode if r1 = A15.

ADR
ADRS

Addition register/register

ADR
ADRS

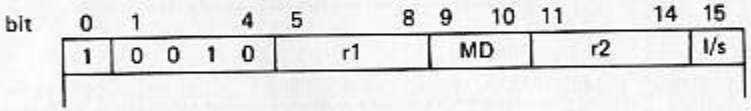
P851M
P852M
P856M
P857M

Syntax: [label] ␣ ADR [*] ␣ r1, r2
[label] ␣ ADRS ␣ r1, r2

The contents of the register specified by r1 are added either to the contents of the register specified by r2 (direct addressing), in which case the sum is always placed in the register specified by r1, or to the contents of the memory address indicated in the register specified by r2 (indirect addressing). In that case the sum is placed either in the register specified by r1 (the l/s indicator being 0) or in the memory address (l/s = 1).

Type	*Function*	*MD*	*l/s*	*Syntax*
T1	(r1) + (r2) → r1	00	n.s.	ADR r1, r2
T3	(r1) + ((r2)) → r1	01	0	ADR* r1, r2
T3	(r1) + ((r2)) → (r2)	01	1	ADRS r1, r2

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



Remarks:
* When l/s = 1 (store), r1 must be ≠ 0.
* Restricted to system mode if r1 = A15.

AD
ADS

Addition

AD
ADS

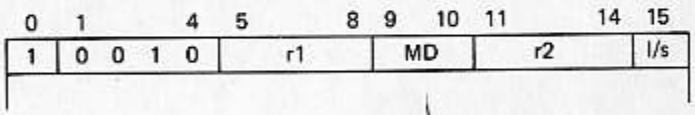
P851M
P852M
P856M
P857M

Syntax: [label] ␣ AD [S] [*] ␣ r1, m[, r2]

The contents of the effective memory address are added to the contents of the register specified by r1.
The sum is placed either in the register specified by r1, in which case the load/store must be 0, or in the effective memory address when the load/store indicator is 1.

Type	*Function*	*MD*	*l/s*	*Syntax*
T4	(r1) + (m) → r1	10	0	AD r1, m
T4	(r1) + (m) → m	10	1	ADS r1, m
T5	(r1) + (m + (r2)) → r1	10	0	AD r1, m, r2
T5	(r1) + (m + (r2)) → m + (r2)	10	1	ADS r1, m, r2
T6	(r1) + ((m)) → r1	11	0	AD* r1, m
T6	(r1) + ((m)) → (m)	11	1	ADS* r1, m
T7	(r1) + ((m + (r2))) → r1	11	0	AD* r1, m, r2
T7	(r1) + ((m + (r2))) → (m + (r2))	11	1	ADS* r1, m, r2

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



Remarks:
* When l/s = 1, r1 must be ≠ 0.
* Restricted to system mode if r1 = A15.

IMR	*Increment memory/register*	IMR	P851M P852M P856M P857M

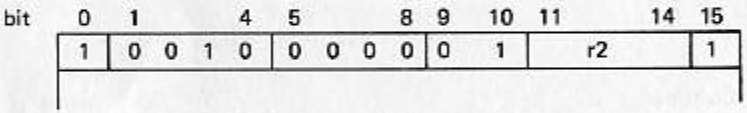
Syntax: [label] ␣ IMR ␣ r2

The contents of the effective memory address indicated in the register specified by r2 (indirect) are increased by one.

Type	*Function*	*Syntax*
T3	((r2)) + 1 → (r2)	IMR r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



IM	*Increment memory*	IM	P851M P852M P856M P857M

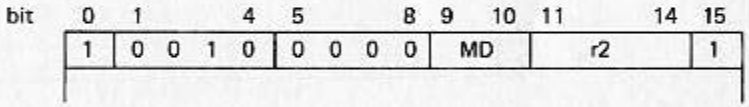
Syntax: [label] ␣ IM [*] ␣ m [, r2]

This instruction increases by 1 the contents of the effective memory address, after which the value of the effective memory address is replaced by the new value.

Type	*Function*	*MD*	*Syntax*
T4	(m) + 1 → m	10	IM m
T5	(m + (r2)) + 1 → m + (r2)	10	IM m, r2
T6	((m)) + 1 → (m)	11	IM* m
T7	((m + (r2))) + 1 → (m + (r2))	11	IM* m, r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



SUK SUKL	*Subtract constant*	SUK SUKL	P851M P852M P856M P857M

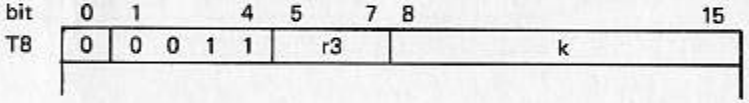
Syntax:
[label] ␣ SUK␣ r3, k — T8
[label] ␣ SUKL ␣ r1, lk — T2

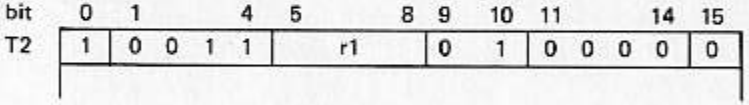
T8 The positive constant k is subtracted from the contents of the register specified in r3. The result is placed in r3.

T2 The positive or negative constant lk is subtracted from the contents of the register specified in r1. The result is placed in r1.

Type	*Function*	*Syntax*
T8	(r3) − k → r3	SUK r3, k
T2	(r1) − lk → r1	SUKL r1, lk

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow





Remark:
Restricted to system mode if r1 = A15.

SUR SURS	*Subtract register/register*	SUR SURS	P851M P852M P856M P857M

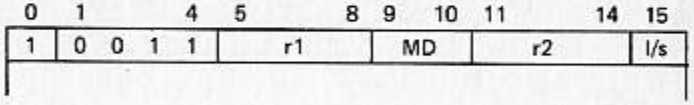
Syntax:
[label] ␣ SUR [*] ␣ r1, r2
[label] ␣ SURS ␣ r1, r2

The contents of the register specified by r2 (direct addressing) or the contents of the memory address indicated in the register specified by r2 (indirect addressing) are subtracted from the contents of the 16-bit register specified by r1. The result of this operation is placed:

− (direct addressing) : in the register specified by r1
− (indirect addressing) : either in the register specified by r1 (l/s = 0) in the memory address indicated in the register specified by r2 (l/s = 1).

Type	*Function*	*MD*	*l/s*	*Syntax*
T1	(r1) − (r2) → r1	00	0	SUR r1, r2
T3	(r1) − ((r2)) → r1	01	0	SUR* r1, r2
T3	(r1) − ((r2)) → (r2)	01	1	SURS r1, r2

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



Remark:
* When l/s = 1, r1 must be ≠ 0
* Restricted to system mode if r1 = A15.

SU
SUS

Subtract word

SU
SUS

P851M
P852M
P856M
P857M

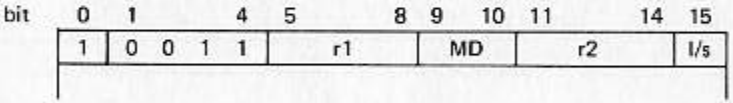
Syntax: [label]␣ SU[S] [*]␣ r1, m[, r2]

The contents of the effective memory address are subtracted from the contents of the register specified by r1. The result is placed in the register specified by r1, when the l/s bit is 0, or in the effective memory address when l/s is 1.

Type	*Function*	*MD*	*l/s*	*Syntax*	
T4	(r1) – (m) → r1	10	0	SU	r1, m
T4	(r1) – (m) → m	10	1	SUS	r1, m
T5	(r1) – (m + (r2)) → r1	10	0	SU	r1, m, r2
T5	(r1) – (m + (r2)) → m + (r2)	10	1	SUS	r1, m, r2
T6	(r1) – ((m)) → r1	11	0	SU*	r1, m
T6	(r1) – ((m)) → (m)	11	1	SUS*	r1, m
T7	(r1) – ((m + (r2))) → r1	11	0	SU*	r1, m, r2
T7	(r1) – ((m + (r2))) → (m + (r2))	11	1	SUS*	r1, m, r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



Remark:
* When the l/s bit = 1, r1 must be ≠ 0
* Restricted to system mode if r1 = A15.

CWK

Compare word with constant

CWK

P851M
P852M
P856M
P857M

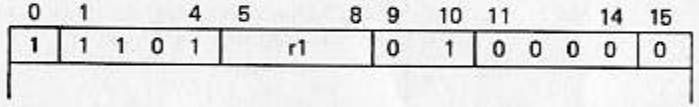
Syntax: [label] ␣ CWK ␣ r1, lk

The contents of the register specified by r1 are compared with the constant. The result of this comparison is stored in the condition register.

Type	*Function*	*Syntax*
T2	(r1) ↔ lk → CR	CWK r1, lk

Condition register:

CR = 0 if (r1) = lk
1 if (r1) > lk
2 if (r1) < lk



Remark:
Restricted to system mode if r1 = A15.

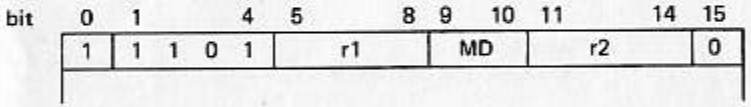
CWR *Compare words register/register* **CWR** P851M P852M P856M P857M

Syntax: [label] ␣ CWR [*] ␣ r1, r2

The contents of the 16-bit register specified by r1 are compared with the contents of the 16-bit register specified by r2 (direct addressing) or with the contents of the memory address held in the register specified by r2 (indirect addressing).
The result of the comparison is stored in the condition register.

Type	*Function*	*MD*	*I/s*	*Syntax*
T1	(r1) ↔ (r2) → CR	00	0	CWR r1, r2
T3	(r1) ↔ ((r2)) → CR	01	0	CWR* r1, r2

Condition register:
CR = 0 if (r1) = (2nd operand)
1 if (r1) > (2nd operand)
2 if (r1) < (2nd operand)



Remark:
Restricted to system mode if r1 = A15.

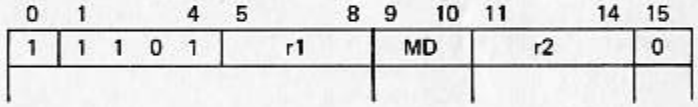
CW *Compare words* **CW** P851M P852M P856M P857M

Syntax: [label] ␣ CW [*] ␣ r1, m [, r2]

The contents of the 16-bit register specified by r1 are compared with the contents of the effective memory address which is found in the word following the instruction.
The result of this comparison is stored in the condition register.

Type	*Function*	*MD*	*Syntax*
T4	(r1) ↔ (m) → CR	10	CW r1, m
T5	(r1) ↔ (m + (r2)) → CR	10	CW r1, m, r2
T6	(r1) ↔ ((m)) → CR	11	CW* r1, m
T7	(r1) ↔ ((m + (r2))) → CR	11	CW* r1, m, r2

Condition register:
CR = 0 if (r1) = 2nd operand
1 if (r1) > 2nd operand
2 if (r1) < 2nd operand



Remark:
Restricted to system mode if r1 = A15.

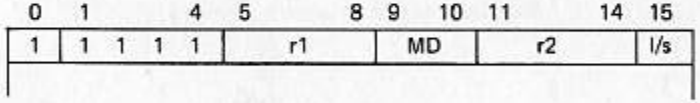
C1 C1S	*Ones complement*	C1 C1S	P851M P852M P856M P857M

Syntax: [label] ␣ C1 [*]␣ r1, m [, r2]
[label] ␣ C1S [*]␣ m [, r2]

Logic Complement: One bits in the specified word or register become 0 and vice versa. The logic complement of the effective memory address replaces either the contents of the 16-bit register specified by r1 or the contents of the effective memory address, depending on the state of the I/s indicator.

Type	*Function*		*MD*	*I/s*	*Syntax*	
T4	$\overline{(m)}$	→ r1	10	0	C1	r1, m
T4	$\overline{(m)}$	→ m	10	1	C1S	m
T5	$\overline{(m + (r2))}$	→ r1	10	0	C1	r1, m, r2
T5	$\overline{(m + (r2))}$	→ m + (r2)	10	1	C1S	m, r2
T6	$\overline{((m))}$	→ r1	11	0	C1*	r1, m
T6	$\overline{((m))}$	→ (m)	11	1	C1S*	m
T7	$\overline{((m + (r2)))}$	→ r1	11	0	C1*	r1, m, r2
T7	$\overline{((m + (r2)))}$	→ (m + (r2))	11	1	C1S*	m, r2

Condition register: CR = 0 if result = 0
1 if result > 0
2 if result < 0



Remark:
* When I/s = 0, r1 must be = 0
* Restricted to system mode when r1 = A15.

C1R C1RS	*Ones complement register/register*	C1R C1RS	P851M P852M P856M P857M

Syntax: [label]␣ C1R [*]␣ r1, r2
[label]␣ C1RS ␣ r2

Logic complement: Bits which contained 1 in the specified register become 0, and vice versa.

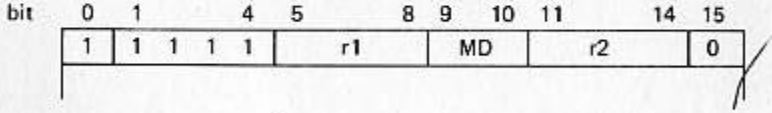
The logic complement of the contents of the 16-bit register specified by r2 (direct addressing) or the contents of the memory address indicated in the register specified by r2 replaces the contents of:

– (direct addressing) : the register specified by r1
– (indirect addressing) : either the register specified by r1 (I/s = 0) or the memory address indicated in the register specified by r2 (I/s = 1).

If r1 is not specified, the default value will be P.

Type	*Function*	*MD*	*I/s*	*Syntax*	
T1	$\overline{(r2)}$ → r1	00	0	C1R	r1, r2
T3	$\overline{((r2))}$ → r1	01	0	C1R*	r1, r2
T3	$\overline{((r2))}$ → (r2)	01	1	C1RS	r2

Condition register: CR = 0 if result = 0
1 if result > 0
2 if result < 0



Remark:
* When I/s = 0, r1 must be = 0
* Restricted to system mode when r1 = A15.

NGR *Negate register* **NGR** P851M P852M P856M P857M

Syntax: [label]␣ NGR ␣ r1, r2

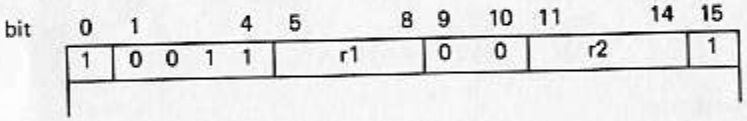
Twos complement.
Zero bits become one and vice versa, +1.

The twos complement of the contents of the register specified by r2 replaces the contents of the register specified by r1.

Type	*Function*	*Syntax*
T1	0 – (r2) → r1	NGR r1, r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



Remark:
* r1 must be ≠ 0
* Restricted to system mode when r1 = A15 (not for P851M).

C2R *Twos complement/register* **C2R** P851M P852M P856M P857M

Syntax: [label]␣ C2R ␣ r2

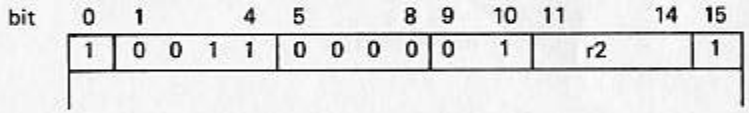
Twos complement.
Zero bits become one and vice versa, +1.

The twos complement (or negative) of the contents of the effective memory address replaces the old contents of this address.

Type	*Function*	*Syntax*
T3	0 – ((r2)) → (r2)	C2R r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



C2 — *Twos complement* — **C2**

P851M
P852M
P856M
P857M

Syntax: [label] ␣ C2 [*] ␣ m [, r2]

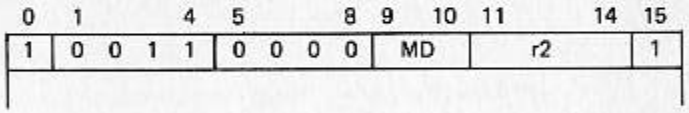
Twos complement.
Zero bits become one and vice versa, +1.

The twos complement (or negative) of the contents of the effective memory address, indicated by the word following the instruction, replaces the old contents.

Type	*Function*	*MD*	*Syntax*
T4	0 – (m) → m	10	C2 m
T5	0 – (m + (r2)) → m + (r2)	10	C2 m, r2
T6	0 – ((m)) → (m)	11	C2* m
T7	0 – ((m + (r2))) → (m + (r2))	11	C2* m, r2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



CMR — *Clear memory/register* — **CMR**

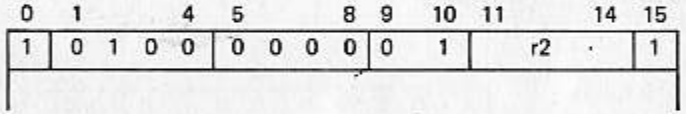
P851M
P852M
P856M
P857M

Syntax: [label] ␣ CMR ␣ r2

The contents of the memory address specified in the register specified by r2 are reset to 0.

Type	*Function*	*Syntax*
T3	0 → (r2)	CMR r2

Condition register: Unchanged



CM

Clear memory

CM

P851M
P852M
P856M
P857M

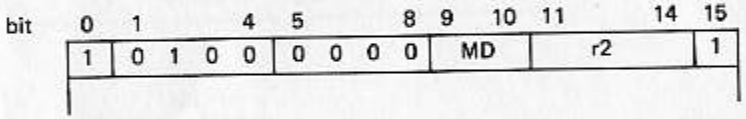
Syntax: [label]␣ CM [*] ␣ m [, r2]

The contents of the effective memory address are reset to 0.

Type	*Function*
T4	0 → m
T5	0 → m + (r2)
T6	0 → (m)
T7	0 → (m + (r2))

MD	*Syntax*
10	CM m
10	CM m, r2
11	CM* m
11	CM* m, r2

Condition register: Unchanged



MUK

Multiply with constant

MUK

P851M
P852M
P856M
P857M

(softw. sim)

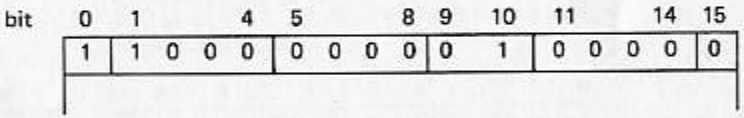
Syntax: [label]␣ MUK ␣ lk

The constant lk is multiplied by the constant of register A2. The result of the multiplication is loaded as a 31-bit product in registers A1 and A2. Bit 0 of A2 is reset to zero. The sign bit of A1 is the sign of the result. Overflow occurs if the result $> 2^{30}-1$.
In that case the two registers contain only the 30 least significant bits while the sign bit may or may not be correct.

Type	*Function*
T2	(A2) x lk → A1, A2

Condition register:

CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



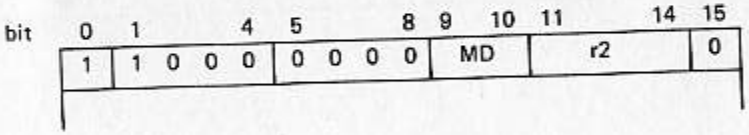
MUR *Multiply register/register* **MUR** P851M P852M P856M P857M (softw.

Syntax: [label]␣ MUR[*]␣ r2

The contents of the register specified by r2 (direct addressing), or the contents of the memory address indicated in r2 (indirect addressing) are multiplied by the contents of A2. The result is loaded as a 31-bit product in A1, A2. The most significant bit of A2 is reset to zero. The sign of the product is stored in the sign bit of A1.
Overflow occurs if the result $> 2^{30} - 1$.
In that case the two registers contain only the 30 least significant bits while the sign bit may or may not be correct.

Type	*Function*		*MD*	*Syntax*	
T1	(A2) x (r2)	→ A1, A2	00	MUR	r2
T3	(A2) x ((r2))	→ A1, A2	01	MUR*	r2

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow



MU *Multiply* **MU** P851M P852M P856M P857M (softw. sim)

Syntax: [label]␣ MU[*]␣ m[, r2]

The contents of register A2 are multiplied by the contents of the effective memory address. The result of this multiplication is loaded as a 31-bit product in registers A1, A2. The most significant bit of A2 is reset to zero. The sign of the product is stored in the sign bit of register A1.
Overflow occurs if result $> 2^{30} - 1$.
In that case the two registers contain only the 30 least significant bits while the sign bit may or may not be correct.

Type	*Function*		*MD*	*Syntax*	
T4	(A2) x (m)	→ A1, A2	10	MU	m
T5	(A2) x (m + (r2))	→ A1, A2	10	MU	m, r2
T6	(A2) x ((m))	→ A1, A2	11	MU*	m
T7	(A2) x ((m + (r2)))	→ A1, A2	11	MU*	m, r2

Condition register:
CR = 0 if result = 0
1 if result > 0
2 if result < 0
3 in case of overflow