Instructions

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
<operand> ← |<operand>|
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	01		AAA			DD		11001010 (ca)								ABS.s rd	1	1
1	01		BBB			DD		10001000 (88)								ABS.s rd	1	1
1	01		XXX			DD		11010110 (d6)								ABS.s addr16, ri	2	4[a]
1	01		ZZZ			DD		11000110 (c6)								ABS.s addr16, ri	2	4[a]
1	01		XXX			DD		11011110 (de)								ABS.s addr32, ri	3	5[a]
1	01		ZZZ			DD		11001110 (ce)								ABS.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

```
000:  X0  001:  X1  010:  X2  011:  X3
100:  A0  101:  A1  110:  A2  111:  A3
```

BBB: Destination register rd

```
000:  Y0  001:  Y1  010:  Y2  011:  Y3
100:  -   101:  Z   110:  SP  111:  -
```

DD: Size s

```
00:  b 8  01:  w 16  10:  l 32  11:  -
```

XXX: Index register ri

```
000:  X0  001:  X1  010:  X2  011:  X3
100:  A0  101:  A1  110:  A2  111:  A3
```

ZZZ: Index register ri

```
000:  0   001:  Z   010:  SP  011:  PC
100:  Y0  101:  Y1  110:  Y2  111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	•

```
rd, C ← rd + <operand> + C
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		01111100 (7c)								ADC.s rd, rs	1	1
0	BB		SSS			DD		01100011 (63)								ADC.s rd, rs	1	1
0	CC		SSS			DD		01110011 (73)								ADC.s rd, rs	1	1
0	AAAA				0	DD		01101001 (69)								ADC.s rd, #imm	2^a	2^b
0	AA		XXX			DD		01110101 (75)								ADC.s rd, addr16, ri	2	3^b
0	AA		ZZZ			DD		01100101 (65)								ADC.s rd, addr16, ri	2	3^b
0	AA		XXX			DD		01111101 (7d)								ADC.s rd, addr32, ri	3	4^b
0	BB		XXX			DD		01101111 (6f)								ADC.s rd, addr32, ri	3	4^b
0	CC		XXX			DD		01111111 (7f)								ADC.s rd, addr32, ri	2	4^b
0	AA		YYY			DD		01111001 (79)								ADC.s rd, addr32, ri	3	4^b
0	BB		YYY			DD		01101011 (6b)								ADC.s rd, addr32, ri	3	4^b
0	CC		YYY			DD		01111011 (7b)								ADC.s rd, addr32, ri	2	4^b
0	AA		ZZZ			DD		01101101 (6d)								ADC.s rd, addr32, ri	3	4^b
0	AA		XXX			DD		01100001 (61)								ADC.s rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		01110001 (71)								ADC.s rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	•

```
rd, C ← rd + <operand>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	AA		SSS			DD		0	1	1	1	1	0	0		ADD.s rd, rs	1	1
1	BB		SSS			DD		0	1	1	0	0	0	1	1	ADD.s rd, rs	1	1
1	CC		SSS			DD		0	1	1	1	0	0	1	1	ADD.s rd, rs	1	1
1	AAAA				0	DD		0	1	1	0	1	0	0	1	ADD.s rd, #imm	2^a	2^b
1	AA		XXX			DD		0	1	1	1	0	1	0	1	ADD.s rd, addr16, ri	2	3^b
1	AA		ZZZ			DD		0	1	1	0	0	1	0	1	ADD.s rd, addr16, ri	2	3^b
1	AA		XXX			DD		0	1	1	1	1	1	0	1	ADD.s rd, addr32, ri	3	4^b
1	BB		XXX			DD		0	1	1	0	1	1	1	1	ADD.s rd, addr32, ri	3	4^b
1	CC		XXX			DD		0	1	1	1	1	1	1	1	ADD.s rd, addr32, ri	2	4^b
1	AA		YYY			DD		0	1	1	1	1	0	0	1	ADD.s rd, addr32, ri	3	4^b
1	BB		YYY			DD		0	1	1	0	1	0	1	1	ADD.s rd, addr32, ri	3	4^b
1	CC		YYY			DD		0	1	1	1	1	0	1	1	ADD.s rd, addr32, ri	2	4^b
1	AA		ZZZ			DD		0	1	1	0	1	1	0	1	ADD.s rd, addr32, ri	3	4^b
1	AA		XXX			DD		0	1	1	0	0	0	0	1	ADD.s rd, (addr16, ri)	2	5^b
1	AA		YYY			DD		0	1	1	1	0	0	0	1	ADD.s rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
rd ← rd ∧ <operand>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		0	0	1	1	1	1	0	0	AND.s rd, rs	1	1
0	BB		SSS			DD		0	0	1	0	0	0	1	1	AND.s rd, rs	1	1
0	CC		SSS			DD		0	0	1	1	0	0	1	1	AND.s rd, rs	1	1
0	AAAA				0	DD		0	0	1	0	1	0	0	1	AND.s rd, #imm	2^a	2^b
0	AA		XXX			DD		0	0	1	1	0	1	0	1	AND.s rd, addr16, ri	2	3^b
0	AA		ZZZ			DD		0	0	1	0	0	1	0	1	AND.s rd, addr16, ri	2	3^b
0	AA		XXX			DD		0	0	1	1	1	1	0	1	AND.s rd, addr32, ri	3	4^b
0	BB		XXX			DD		0	0	1	0	1	1	1	1	AND.s addr32, Ri	3	4^b
0	CC		XXX			DD		0	0	1	1	1	1	1	1	AND.s addr32, Ri	3	4^b
0	AA		YYY			DD		0	0	1	1	1	0	0	1	AND.s rd, addr32, ri	3	4^b
0	BB		YYY			DD		0	0	1	0	1	0	1	1	AND.s addr32, Ri	3	4^b
0	CC		YYY			DD		0	0	1	1	1	0	1	1	AND.s addr32, Ri	3	4^b
0	AA		ZZZ			DD		0	0	1	0	1	1	0	1	AND.s rd, addr32, ri	3	4^b
0	AA		XXX			DD		0	0	1	0	0	0	0	1	AND.s rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		0	0	1	1	0	0	0	1	AND.s rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

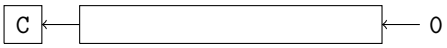
```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	0		RRRR			DD		0	0	0	0	1	0	1	0	ASL.s rd	1	1
0	0		RRRR			DD		0	0	0	0	0	0	1	0	ASL.s rd, #imm	2	2
0	1		RRRR			DD		0	0	0	0	0	0	1	0	ASL.s rd, rs[b]	2	2
0	NN		XXX			DD		0	0	0	1	0	1	1	0	ASL.s addr16, ri [, #N]	2	4[a]
0	NN		ZZZ			DD		0	0	0	0	0	1	1	0	ASL.s addr16, ri [, #N]	2	4[a]
0	NN		XXX			DD		0	0	0	1	1	1	1	0	ASL.s addr32, ri [, #N]	3	5[a]
0	NN		ZZZ			DD		0	0	0	0	1	1	1	0	ASL.s addr32, ri [, #N]	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  8   01:  16   10:  32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

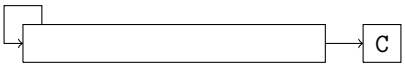
```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	0		RRRR			DD		01001010 (4a)								`ASR.s rd, rs`	1	1
1	0		RRRR			DD		01000010 (42)								`ASR.s rd, #imm`	2	2
1	1		RRRR			DD		01000010 (42)								`ASR.s rd, rs`[b]	2	2
1	NN		XXX			DD		01010110 (56)								`ASR.s addr16, ri [, #N]`	2	4[a]
1	NN		ZZZ			DD		01000110 (46)								`ASR.s addr16, ri [, #N]`	2	4[a]
1	NN		XXX			DD		01011110 (5e)								`ASR.s addr32, ri [, #N]`	3	5[a]
1	NN		ZZZ			DD		01001110 (4e)								`ASR.s addr32, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
if condition
 if link, mem[SP--] ← PC
 addr ← rs + offset
 if indirect, addr ← mem[addr]
 PC ← addr
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
C	D	0	L	RRRR				CCC10000								`Bcc[.l] offset, rs`	2^a	2^b
C	D	1	L	RRRR				CCC10000								`Bcc[.l] (offset, rs)`	2^a	4^b

[a] add 1 word for 32 bit offset (`D` = 1)
[b] add 2 cycles for link (`L` = 1)

`CCCC`: Condition

0000	BPL	¬N
0001	BMI	N
0010	BVC	¬V
0011	BVS	V
0100	BCC	¬C
0101	BCS	C
0110	BNE	¬Z
0111	BEQ	Z
1000	BGE	(N ∧ V) ∨ (¬ N ∧ ¬ V)
1001	BLT	(N ∧ ¬ V) ∨ (¬ N ∧ V)
1010	BLE	Z ∨ (N ∧ ¬ V) ∨ (¬ N ∧ V)
1011	BGT	¬ Z ∧ ((N ∧ V) ∨ (¬ N ∧ ¬ V))
1100	BLS	C ∨ Z
1101	BHI	¬ C ∧ ¬ Z
1110	BNV	false
1111	BRA	true

`D`: Offset size

0: 16 bit 1: 32 bit

`L`: Link

0: no link 1: link

`RRRR`: Base register `rs`

0000:	PC	0001:	SP	0010:	Z	0011:	-
0100:	Y3	0101:	Y2	0110:	Y1	0111:	Y0
1000:	X3	1001:	X2	1010:	X1	1011:	X0
1100:	A3	1101:	A2	1110:	A1	1111:	A0

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

rd ← rd ∧ ¬<operand>

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	AA		SSS			DD		00111100 (3c)								BIC.s rd, rs	1	1
1	BB		SSS			DD		00100011 (23)								BIC.s rd, rs	1	1
1	CC		SSS			DD		00110011 (33)								BIC.s rd, rs	1	1
1	AAAA				0	DD		00101001 (29)								BIC.s rd, #imm	2^a	2^b
1	AA		XXX			DD		00110101 (35)								BIC.s rd, addr16, ri	2	3^b
1	AA		ZZZ			DD		00100101 (25)								BIC.s rd, addr16, ri	2	3^b
1	AA		XXX			DD		00111101 (3d)								BIC.s rd, addr32, ri	3	4^b
1	BB		XXX			DD		00101111 (2f)								BIC.s addr32, Ri	3	4^b
1	CC		XXX			DD		00111111 (3f)								BIC.s addr32, Ri	3	4^b
1	AA		YYY			DD		00111001 (39)								BIC.s rd, addr32, ri	3	4^b
1	BB		YYY			DD		00101011 (2b)								BIC.s addr32, Ri	3	4^b
1	CC		YYY			DD		00111011 (3b)								BIC.s addr32, Ri	3	4^b
1	AA		ZZZ			DD		00101101 (2d)								BIC.s rd, addr32, ri	3	4^b
1	AA		XXX			DD		00100001 (21)								BIC.s rd, (addr16, ri)	2	5^b
1	AA		YYY			DD		00110001 (31)								BIC.s rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

00: A0 01: A1 10: A2 11: A3

AAAA: Destination register rd

0000: A0 0001: A1 0010: A2 0011: A3
0100: X0 0101: X1 0110: X2 0111: X3
1000: Y0 1001: Y1 1010: Y2 1011: Y3
1100: - 1101: Z 1110: SP 1111: -

BB: Destination register rd

00: X0 01: X1 10: X2 11: X3

CC: Destination register rd

00: Y0 01: Y1 10: Y2 11: Y3

DD: Size s

00: b 8 01: w 16 10: l 32 11: -

SSS: Source register rs

000: A0 001: A1 010: A2 011: A3
100: X0 101: X1 110: X2 111: X3

XXX: Index register ri

000: X0 001: X1 010: X2 011: X3
100: A0 101: A1 110: A2 111: A3

YYY: Index register ri

000: Y0 001: Y1 010: Y2 011: Y3
100: 0 101: Z 110: SP 111: PC

ZZZ: Index register ri

000: 0 001: Z 010: SP 011: PC
100: Y0 101: Y1 110: Y2 111: Y3

Operation

N	Z	C	I	D	V
•	•	–	–	–	•

```
 N ← <operand>[N-1]
V ← <operand>[N-2]
Z ← <operand> ∧ rd = 0
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		00110100 (34)								BIT.s rd, rs	1	1
0	AA		ZZZ			DD		00100100 (24)								BIT.s rd, addr16, ri	2	3[a]
0	AA		XXX			DD		00101100 (2c)								BIT.s rd, addr32, ri	3	4[a]
0	AA		ZZZ			DD		00111010 (3a)								BIT.s rd, addr32, ri	3	4[a]

[a] add 1 cycle for 32 bit (s = 2)

AA: Test register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

DD: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
 mem[SP--] ← PC
mem[SP--] ← P
PC ← mem[$fffe - 2*vector]
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
W	NNN			VVVV				00000000 (00)								BRK[.w] #v, #n	2	5

W: Wait for interrupt

```
000:  don't wait  001:  wait
```

NNN: Interrupt number

```
000:  IRQ0  001:  IRQ1  010:  IRQ2  011:  IRQ3
100:  IRQ4  101:  IRQ5  110:  IRQ6  111:  IRQ7
```

VVVV: Vector address

```
0000:  $fffe  0001:  $fffc  0010:  $fffa  0011:  $fff8
0100:  $fff6  0101:  $fff4  0110:  $fff2  0111:  $fff0
1000:  $ffee  1001:  $ffec  1010:  $ffea  1011:  $ffe8
1100:  $ffe6  1101:  $ffe4  1110:  $ffe2  1111:  $ffe0
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

`<operand>` ← `<operand>` ∧ ¬ `mask`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0		AAA				RRRR					00011000 (18)					`CBT rd, #imm`	1	1
0		AAA				XXXX					00010010 (12)					`CBT addr32, ri, #imm`	3	5
0		BBB				RRRR					00011000 (58)					`CBT rd, #imm`	1	1
0		BBB				XXXX					00010010 (52)					`CBT addr32, ri, #imm`	3	5
0		CCC				RRRR					00011000 (d8)					`CBT rd, #imm`	1	1
0		CCC				XXXX					00010010 (d2)					`CBT addr32, ri, #imm`	3	5
0		DDD				RRRR					00011000 (b8)					`CBT rd, #imm`	1	1
0		DDD				XXXX					00010010 (92)					`CBT addr32, ri, #imm`	3	5

`AAA`: Bit number

000:	0	001:	7	010:	8	011:	15
100:	16	101:	23	110:	24	111:	31

`BBB`: Bit number

000:	2	001:	5	010:	10	011:	13
100:	18	101:	21	110:	26	111:	29

`CCC`: Bit number

000:	3	001:	4	010:	11	011:	12
100:	19	101:	20	110:	27	111:	28

`DDD`: Bit number

000:	6	001:	1	010:	14	011:	9
100:	22	101:	17	110:	30	111:	25

`RRRR`: Operand register `rd`

0000:	P	0001:	Z	0010:	SP	0011:	-
0100:	Y0	0101:	Y1	0110:	Y2	0111:	Y3
1000:	X0	1001:	X1	1010:	X2	1011:	X3
1100:	A0	1101:	A1	1110:	A2	1111:	A3

`XXXX`: Index register `ri`

0000:	A0	0001:	A1	0010:	A2	0011:	A3
0100:	X0	0101:	X1	0110:	X2	0111:	X3
1000:	Y0	1001:	Y1	1010:	Y2	1011:	Y3
1100:	P	1101:	Z	1110:	SP	1111:	-

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
<operand> ← <operand> - N
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	NN		AAA			DD		11001010 (ca)								DEC.s rd	1	1
0	NN		BBB			DD		10001000 (88)								DEC.s rd	1	1
0	NN		XXX			DD		11010110 (d6)								DEC.s addr16, ri	2	4[a]
0	NN		ZZZ			DD		11000110 (c6)								DEC.s addr16, ri	2	4[a]
0	NN		XXX			DD		11011110 (de)								DEC.s addr32, ri	3	5[a]
0	NN		ZZZ			DD		11001110 (ce)								DEC.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

BBB: Destination register rd

000:	Y0	001:	Y1	010:	Y2	011:	Y3
100:	-	101:	Z	110:	SP	111:	-

DD: Size

00:	b 8	01:	w 16	10:	l 32	11:	-

XXX: Index register ri

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

ZZZ: Index register ri

000:	0	001:	Z	010:	SP	011:	PC
100:	Y0	101:	Y1	110:	Y2	111:	Y3

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

rd ← rd ⊕ <operand>

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		01011100 (5c)								EOR.s rd, rs	1	1
0	BB		SSS			DD		01000011 (43)								EOR.s rd, rs	1	1
0	CC		SSS			DD		01010011 (53)								EOR.s rd, rs	1	1
0	AAAA				0	DD		01001001 (49)								EOR.s rd, #imm	2^a	2^b
0	AA		XXX			DD		01010101 (55)								EOR.s rd, addr16, ri	2	3^b
0	AA		ZZZ			DD		01000101 (45)								EOR.s rd, addr16, ri	2	3^b
0	AA		XXX			DD		01011101 (5d)								EOR.s rd, addr32, ri	3	4^b
0	BB		XXX			DD		01001111 (4f)								EOR.s rd, addr32, ri	3	4^b
0	CC		XXX			DD		01011111 (5f)								EOR.s rd, addr32, ri	3	4^b
0	AA		YYY			DD		01011001 (59)								EOR.s rd, addr32, ri	3	4^b
0	BB		YYY			DD		01001011 (4b)								EOR.s rd, addr32, ri	3	4^b
0	CC		YYY			DD		01011011 (5b)								EOR.s rd, addr32, ri	3	4^b
0	AA		ZZZ			DD		01001101 (4d)								EOR.s rd, addr32, ri	3	4^b
0	AA		XXX			DD		01000001 (41)								EOR.s rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		01010001 (51)								EOR.s rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

00: A0 01: A1 10: A2 11: A3

AAAA: Destination register rd

0000: A0 0001: A1 0010: A2 0011: A3
0100: X0 0101: X1 0110: X2 0111: X3
1000: Y0 1001: Y1 1010: Y2 1011: Y3
1100: - 1101: Z 1110: SP 1111: -

BB: Destination register rd

00: X0 01: X1 10: X2 11: X3

CC: Destination register rd

00: Y0 01: Y1 10: Y2 11: Y3

DD: Size

00: b 8 01: w 16 10: l 32 11: -

SSS: Source register rs

000: A0 001: A1 010: A2 011: A3
100: X0 101: X1 110: X2 111: X3

XXX: Index register ri

000: X0 001: X1 010: X2 011: X3
100: A0 101: A1 110: A2 111: A3

YYY: Index register ri

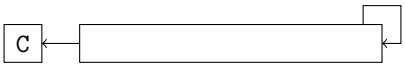
000: Y0 001: Y1 010: Y2 011: Y3
100: 0 101: Z 110: SP 111: PC

ZZZ: Index register ri

000: 0 001: Z 010: SP 011: PC
100: Y0 101: Y1 110: Y2 111: Y3

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	0		RRRR			DD		00001010 (0a)								`ESL.s rd`	1	1
1	0		RRRR			DD		00000010 (02)								`ESL.s rd, #imm`	2	2
1	1		RRRR			DD		00000010 (02)								`ESL.s rd, rs`[b]	2	2
1	NN		XXX			DD		00010110 (16)								`ESL.s addr16, ri [, #N]`	2	4[a]
1	NN		ZZZ			DD		00000110 (06)								`ESL.s addr16, ri [, #N]`	2	4[a]
1	NN		XXX			DD		00011110 (1e)								`ESL.s addr32, ri [, #N]`	3	5[a]
1	NN		ZZZ			DD		00001110 (0e)								`ESL.s addr32, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  8   01:  16   10:  32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
<operand> ← Extend(<operand>)
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	10		AAA			DD		11001010 (ca)								EXT.s rd	1	1
1	10		BBB			DD		10001000 (88)								EXT.s rd	1	1
1	10		XXX			DD		11010110 (d6)								EXT.s addr16, ri	2	4[a]
1	10		ZZZ			DD		11000110 (c6)								EXT.s addr16, ri	2	4[a]
1	10		XXX			DD		11011110 (de)								EXT.s addr32, ri	3	5[a]
1	10		ZZZ			DD		11001110 (ce)								EXT.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

BBB: Destination register rd

000:	Y0	001:	Y1	010:	Y2	011:	Y3
100:	-	101:	Z	110:	SP	111:	-

XXX: Index register ri

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

ZZZ: Index register ri

000:	0	001:	Z	010:	SP	011:	PC
100:	Y0	101:	Y1	110:	Y2	111:	Y3

DD: Size s

00: b 1 → 8 01: w 8 → 16 10: 16 → 32 11: -

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
<operand> ← <operand> + N
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	NN		AAA			DD		11001010 (e8)								INC.s rd	1	1
0	NN		BBB			DD		10001000 (c8)								INC.s rd	1	1
0	NN		XXX			DD		11010110 (f6)								INC.s addr16, ri	2	4[a]
0	NN		ZZZ			DD		11000110 (e6)								INC.s addr16, ri	2	4[a]
0	NN		XXX			DD		11011110 (fe)								INC.s addr32, ri	3	5[a]
0	NN		ZZZ			DD		11001110 (ee)								INC.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

BBB: Destination register rd

000:	Y0	001:	Y1	010:	Y2	011:	Y3
100:	-	101:	Z	110:	SP	111:	-

DD: Size

00:	b 8	01:	w 16	10:	l 32	11:	-

XXX: Index register ri

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

ZZZ: Index register ri

000:	0	001:	Z	010:	SP	011:	PC
100:	Y0	101:	Y1	110:	Y2	111:	Y3

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
PC ← <address>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	0000000							01001100 (4c)								`JMP addr32`	3	3
0	0000000							01101100 (6c)								`JMP (addr32)`	3	5

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
 mem[SP--] ← PC
PC ← <address>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	00		SSS			--		01000000 (20)								JSR [(rs),] addr32	3	5

SSS: Stack pointer register `rs`

```
000:  SP  001:  PC  010:  0   011:  Z
100:  Y2  101:  Y3  110:  Y0  111:  Y1
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
rd ← <operand>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AAAA				0	DD		10101001 (a9)								LDR rd, #imm	2^a	2^b
0	AA		XXX			DD		10110101 (b5)								LDR rd, addr16, ri	2	3^b
0	CC		XXX			DD		10110100 (b4)								LDR rd, addr16, ri	2	3^b
0	BB		YYY			DD		10110110 (b6)								LDR rd, addr16, ri	2	3^b
0	AA		ZZZ			DD		10100101 (a5)								LDR rd, addr16, ri	2	3^b
0	CC		ZZZ			DD		10100100 (a4)								LDR rd, addr16, ri	2	3^b
0	BB		ZZZ			DD		10100110 (a6)								LDR rd, addr16, ri	2	3^b
0	AA		XXX			DD		10111101 (bd)								LDR rd, addr32, ri	3	4^b
0	BB		XXX			DD		10101111 (af)								LDR rd, addr32, ri	3	4^b
0	CC		XXX			DD		10111100 (bc)								LDR rd, addr32, ri	3	4^b
0	AA		YYY			DD		10111001 (b9)								LDR rd, addr32, ri	3	4^b
0	BB		YYY			DD		10111110 (be)								LDR rd, addr32, ri	3	4^b
0	BB		ZZZ			DD		10101110 (ae)								LDR rd, addr32, ri	3	4^b
0	CC		ZZZ			DD		10101100 (ac)								LDR rd, addr32, ri	3	4^b
0	AA		ZZZ			DD		10101101 (ad)								LDR rd, addr32, ri	3	4^b
0	AA		XXX			DD		10100001 (a1)								LDR rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		10110001 (b1)								LDR rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

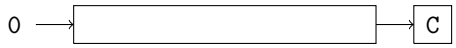
```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	0	RRRR				DD		01001010 (4a)								`LSR.s rd, rs`	1	1
0	0	RRRR				DD		01000010 (42)								`LSR.s rd, #imm`	2	2
0	1	RRRR				DD		01000010 (42)								`LSR.s rd, rs`[b]	2	2
0	NN		XXX			DD		01010110 (56)								`LSR.s addr16, ri [, #N]`	2	4[a]
0	NN		ZZZ			DD		01000110 (46)								`LSR.s addr16, ri [, #N]`	2	4[a]
0	NN		XXX			DD		01011110 (5e)								`LSR.s addr32, ri [, #N]`	3	5[a]
0	NN		ZZZ			DD		01001110 (4e)								`LSR.s addr32, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	•

```
rd ← rs
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
rd XXX			rs AAA			DD		10101010 (aa)								MOV rd, rs	1	1
rd YYY			rs AAA			DD		10101000 (a8)								MOV rd, rs	1	1
rd XXX			rs SSS			DD		10111010 (ba)								MOV rd, rs	1	1
rd AAA			rs XXX			DD		10001010 (8a)								MOV rd, rs	1	1
rd SSS			rs XXX			DD		10001010 (8a)								MOV rd, rs	1	1
rd AAA			rs YYY			DD		10011000 (98)								MOV rd, rs	1	1

DD: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

AAA: Source/destination register **rs/rd**

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

XXX: Source/destination register **rs/rd**

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  P    101:  Z    110:  SP   111:  -
```

YYY: Source/destination register **rs/rd**

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  A0   101:  A1   110:  A2   111:  A3
```

SSS: Source/destination register **rs/rd**

```
000:  SP   001:  -    010:  P    011:  Z
100:  X2   101:  X3   110:  X0   111:  X1
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

`<operand>` ← `-<operand>`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	00		AAA			DD		11001010 (ca)								NEG.s rd	1	1
1	00		BBB			DD		10001000 (88)								NEG.s rd	1	1
1	00		XXX			DD		11010110 (d6)								NEG.s addr16, ri	2	4[a]
1	00		ZZZ			DD		11000110 (c6)								NEG.s addr16, ri	2	4[a]
1	00		XXX			DD		11011110 (de)								NEG.s addr32, ri	3	5[a]
1	00		ZZZ			DD		11001110 (ce)								NEG.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

BBB: Destination register rd

000:	Y0	001:	Y1	010:	Y2	011:	Y3
100:	-	101:	Z	110:	SP	111:	-

DD: Size s

00:	b 8	01:	w 16	10:	l 32	11:	-

XXX: Index register ri

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

ZZZ: Index register ri

000:	0	001:	Z	010:	SP	011:	PC
100:	Y0	101:	Y1	110:	Y2	111:	Y3

Operation

N	Z	C	I	D	V
—	—	—	—	—	—

```
PC ← PC + N
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
NNNNNNNN								11101010 (ea)								NOP [#n]	n+1	1

NNNNNNNN: Number of program words to skip

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

`<operand>` ← ¬`<operand>`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	11		AAA			DD		11001010 (ca)								NOT.s rd	1	1
1	11		BBB			DD		10001000 (88)								NOT.s rd	1	1
1	11		XXX			DD		11010110 (d6)								NOT.s addr16, ri	2	4[a]
1	11		ZZZ			DD		11000110 (c6)								NOT.s addr16, ri	2	4[a]
1	11		XXX			DD		11011110 (de)								NOT.s addr32, ri	3	5[a]
1	11		ZZZ			DD		11001110 (ce)								NOT.s addr32, ri	3	5[a]

[a] add 2 cycles for 32 bit (s=2)

AAA: Destination register rd

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

BBB: Destination register rd

000:	Y0	001:	Y1	010:	Y2	011:	Y3
100:	-	101:	Z	110:	SP	111:	-

DD: Size s

00:	b 8	01:	w 16	10:	l 32	11:	-

XXX: Index register ri

000:	X0	001:	X1	010:	X2	011:	X3
100:	A0	101:	A1	110:	A2	111:	A3

ZZZ: Index register ri

000:	0	001:	Z	010:	SP	011:	PC
100:	Y0	101:	Y1	110:	Y2	111:	Y3

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
rd ← rd ∨ <operand>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		0	0	0	1	1	1	0	0	ORA.d rd, rs	1	1
0	BB		SSS			DD		0	0	0	0	0	0	1	1	ORA.d rd, rs	1	1
0	CC		SSS			DD		0	0	0	1	0	0	1	1	ORA.d rd, rs	1	1
0	AAAA				0	DD		0	0	0	0	1	0	0	1	ORA.d rd, #imm	2^a	2^b
0	AA		XXX			DD		0	0	0	1	0	1	0	1	ORA.d rd, addr16, ri	2	3^b
0	AA		ZZZ			DD		0	0	0	0	0	1	0	1	ORA.d rd, addr16, ri	2	3^b
0	AA		XXX			DD		0	0	0	1	1	1	0	1	ORA.d rd, addr32, ri	3	4^b
0	BB		XXX			DD		0	0	0	0	1	1	1	1	ORA.d rd, addr32, ri	3	4^b
0	CC		XXX			DD		0	0	0	1	1	1	1	1	ORA.d rd, addr32, ri	3	4^b
0	AA		YYY			DD		0	0	0	1	1	0	0	1	ORA.d rd, addr32, ri	3	4^b
0	BB		YYY			DD		0	0	0	0	1	0	1	1	ORA.d rd, addr32, ri	3	4^b
0	CC		YYY			DD		0	0	0	1	1	0	1	1	ORA.d rd, addr32, ri	3	4^b
0	AA		ZZZ			DD		0	0	0	0	1	1	0	1	ORA.d rd, addr32, ri	3	4^b
0	AA		XXX			DD		0	0	0	0	0	0	0	1	ORA.d rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		0	0	0	1	0	0	0	1	ORA.d rd, (addr16), ri	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
mem[rs--] ← rd
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		01001000 (48)								PSH.s [(rs),] rd	1	2^a
0	XX		SSS			DD		01000100 (44)								PSH.s [(rs),] rd	1	2^a
0	YY		SSS			DD		01010100 (54)								PSH.s [(rs),] rd	1	2^a
0	PP		SSS			DD		00001000 (08)								PSH.s [(rs),] rd	1	2^a

[a]add 1 cycle for 32 bit (s=2)

AA: Register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

XX: Register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

YY: Register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

PP: Register rd

```
00:  P   01:  Z   10:  SP   11:  PC
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Stack pointer rs

```
000:  SP   001:  PC   010:  0    011:  P
100:  Y2   101:  Y3   110:  Y0   111:  Y1
```

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
rd ← mem[++rs]
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		01101000 (68)								PUL.s [(rs),] rd	1	2^a
0	XX		SSS			DD		01100100 (64)								PUL.s [(rs),] rd	1	2^a
0	YY		SSS			DD		01110100 (74)								PUL.s [(rs),] rd	1	2^a
0	PP		SSS			DD		00101000 (28)								PUL.s [(rs),] rd	1	2^a

[a] add 1 cycle for 32 bit (s=2)

AA: Register rd

```
00: A0  01: A1  10: A2  11: A3
```

XX: Register rd

```
00: X0  01: X1  10: X2  11: X3
```

YY: Register rd

```
00: Y0  01: Y1  10: Y2  11: Y3
```

PP: Register rd

```
00: P  01: Z  10: SP  11: PC
```

DD: Size s

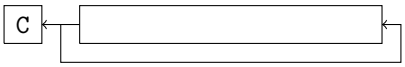
```
00: b 8  01: w 16  10: l 32  11: -
```

SSS: Stack pointer rs

```
000: SP  001: PC  010: 0   011: P
100: Y2  101: Y3  110: Y0  111: Y1
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	0		RRRR			DD		00101010 (2a)								`RLB.s rd`	1	1
1	0		RRRR			DD		00100010 (22)								`RLB.s rd, #n`	2	2
1	1		RRRR			DD		00100010 (22)								`RLB.s rd, rs`[b]	2	2
1	NN		XXX			DD		00110110 (36)								`RLB.s addr16, ri [, #N]`	2	4[a]
1	NN		ZZZ			DD		00100110 (26)								`RLB.s addr16, ri [, #N]`	2	4[a]
1	NN		XXX			DD		00111110 (3e)								`RLB.s addr32, ri [, #N]`	3	5[a]
1	NN		ZZZ			DD		00101110 (2e)								`RLB.s addr32, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size `s`

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

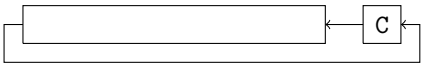
```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	0		RRRR			DD		00101010 (2a)								ROL.s rd	1	1
0	0		RRRR			DD		00100010 (22)								ROL.s rd, #n	2	2
0	1		RRRR			DD		00100010 (22)								ROL.s rd, rs[b]	2	2
0	NN		XXX			DD		00110110 (36)								ROL.s addr16, ri [, #N]	2	4[a]
0	NN		ZZZ			DD		00100110 (26)								ROL.s addr16, ri [, #N]	2	4[a]
0	NN		XXX			DD		00111110 (3e)								ROL.s addr32, ri [, #N]	3	5[a]
0	NN		ZZZ			DD		00101110 (2e)								ROL.s addr32, ri [, #N]	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size `s`

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

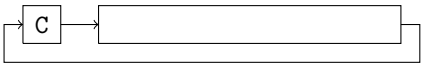
```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	0	RRRR				DD		01101010 (6a)								`ROR.s rd`	1	1
0	0	RRRR				DD		01100010 (62)								`ROR.s rd, #n`	2	2
0	1	RRRR				DD		01100010 (62)								`ROR.s rd, rs`[b]	2	2
0	NN		XXX			DD		01110110 (76)								`ROR.s addr16, ri [, #N]`	2	4[a]
0	NN		ZZZ			DD		01100110 (66)								`ROR.s addr16, ri [, #N]`	2	4[a]
0	NN		XXX			DD		01111110 (7e)								`ROR.s addr16, ri [, #N]`	3	5[a]
0	NN		ZZZ			DD		01101110 (6e)								`ROR.s addr16, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  8   01:  16   10:  32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

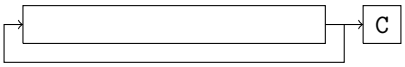
```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	–



Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	0		RRRR			DD		01101010 (6a)								`RRB.s rd`	1	1
1	0		RRRR			DD		01100010 (62)								`RRB.s rd, #n`	2	2
1	1		RRRR			DD		01100010 (62)								`RRB.s rd, rs`[b]	2	2
1	NN		XXX			DD		01110110 (76)								`RRB.s addr16, ri [, #N]`	2	4[a]
1	NN		ZZZ			DD		01100110 (66)								`RRB.s addr16, ri [, #N]`	2	4[a]
1	NN		XXX			DD		01111110 (7e)								`RRB.s addr16, ri [, #N]`	3	5[a]
1	NN		ZZZ			DD		01101110 (6e)								`RRB.s addr16, ri [, #N]`	3	5[a]

[a] add 2 cycles for 32 bit (`s` = 2)
[b] register `rs` specified by bits 0-3 of the following word

`NN`: Shift amount

```
00:  1   01:  2   10:  3   11:  4
```

`DD`: Size

```
00:  8   01:  16   10:  32   11:  -
```

`RRRR`: Register `rd`

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

`XXX`: Index register `ri`

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

`ZZZ`: Index register `ri`

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
 P ← mem[++SP]
PC ← mem[++SP]
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	00		SSS			--		01000000 (40)								`RTI [(rs)]`	1	4

SSS: Stack pointer register `rs`

```
000:  SP  001:  PC  010:  0   011:  Z
100:  Y2  101:  Y3  110:  Y0  111:  Y1
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

```
PC ← mem[++SP]
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	00		SSS			--		01100000 (60)								RTS [(rs)]	1	4

SSS: Stack pointer register `rs`

```
000:  SP   001:  PC   010:  0    011:  Z
100:  Y2   101:  Y3   110:  Y0   111:  Y1
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	•

`rd, C ← rd - <operand> - ¬C`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		SSS			DD		11111100 (fc)								SBC.s rd, rs	1	1
0	BB		SSS			DD		11100011 (e3)								SBC.s rd, rs	1	1
0	CC		SSS			DD		11110011 (f3)								SBC.s rd, rs	1	1
0	AAAA				0	DD		11101001 (e9)								SBC.s rd, #imm	2^a	2^b
0	AA		XXX			DD		11110101 (f5)								SBC.s rd, addr16, ri	2	2^b
0	AA		ZZZ			DD		11100101 (e5)								SBC.s rd, addr16, ri	2	2^b
0	AA		XXX			DD		11111101 (fd)								SBC.s rd, addr32, ri	3	4^b
0	BB		XXX			DD		11101111 (ef)								SBC.s rd, addr32, ri	3	4^b
0	CC		XXX			DD		11111111 (ff)								SBC.s rd, addr32, ri	3	4^b
0	AA		YYY			DD		11111001 (f9)								SBC.s rd, addr32, ri	3	4^b
0	BB		YYY			DD		11101011 (eb)								SBC.s rd, addr32, ri	3	4^b
0	CC		YYY			DD		11111011 (fb)								SBC.s rd, addr32, ri	3	4^b
0	AA		ZZZ			DD		11101101 (ed)								SBC.s rd, addr32, ri	3	4^b
0	AA		XXX			DD		11100001 (e1)								SBC.s rd, (addr16, ri)	2	5^b
0	AA		YYY			DD		11110001 (f1)								SBC.s rd, (addr16, ri)	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

`<operand>` ← `<operand>` ∨ `mask`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AAA			RRRR				00111000 (38)								SBT rd, #imm	1	1
0	AAA			XXXX				00110010 (32)								SBT addr32, ri, #imm	3	5
0	BBB			RRRR				01111000 (78)								SBT rd, #imm	1	1
0	BBB			XXXX				01110010 (72)								SBT addr32, ri, #imm	3	5
0	CCC			RRRR				11111000 (f8)								SBT rd, #imm	1	1
0	CCC			XXXX				11110010 (f2)								SBT addr32, ri, #imm	3	5
0	DDD			RRRR				10011111 (9f)								SBT rd, #imm	1	1
0	DDD			XXXX				10110010 (b2)								SBT addr32, ri, #imm	3	5

AAA: Bit number

000:	0	001:	7	010:	8	011:	15
100:	16	101:	23	110:	24	111:	31

BBB: Bit number

000:	2	001:	5	010:	10	011:	13
100:	18	101:	21	110:	26	111:	29

CCC: Bit number

000:	3	001:	4	010:	11	011:	12
100:	19	101:	20	110:	27	111:	28

DDD: Bit number

000:	6	001:	1	010:	14	011:	9
100:	22	101:	17	110:	30	111:	25

RRRR: Operand register `rd`

0000:	P	0001:	Z	0010:	SP	0011:	-
0100:	Y0	0101:	Y1	0110:	Y2	0111:	Y3
1000:	X0	1001:	X1	1010:	X2	1011:	X3
1100:	A0	1101:	A1	1110:	A2	1111:	A3

XXXX: Index register `ri`

0000:	A0	0001:	A1	0010:	A2	0011:	A3
0100:	X0	0101:	X1	0110:	X2	0111:	X3
1000:	Y0	1001:	Y1	1010:	Y2	1011:	Y3
1100:	0	1101:	Z	1110:	SP	1111:	-

Operation

N	Z	C	I	D	V
•	•	–	–	–	–

```
<operand> ← rd
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
0	AA		XXX			DD		10010101 (95)								STR.s rd, addr16, ri	2	3[a]
0	CC		XXX			DD		10010100 (94)								STR.s rd, addr16, ri	2	3[a]
0	BB		YYY			DD		10010110 (96)								STR.s rd, addr16, ri	2	3[a]
0	AA		ZZZ			DD		10000101 (85)								STR.s rd, addr16, ri	2	3[a]
0	BB		ZZZ			DD		10000110 (86)								STR.s rd, addr16, ri	2	3[a]
0	CC		ZZZ			DD		10000100 (84)								STR.s rd, addr16, ri	2	3[a]
0	AA		XXX			DD		10011101 (9d)								STR.s rd, addr32, ri	3	4[a]
0	BB		XXX			DD		10001111 (8f)								STR.s rd, addr32, ri	3	4[a]
0	CC		XXX			DD		10011100 (9c)								STR.s rd, addr32, ri	3	4[a]
0	AA		YYY			DD		10011001 (99)								STR.s rd, addr32, ri	3	4[a]
0	BB		YYY			DD		10011110 (9e)								STR.s rd, addr32, ri	3	4[a]
0	AA		ZZZ			DD		10001101 (8d)								STR.s rd, addr32, ri	3	4[a]
0	BB		ZZZ			DD		10001110 (8e)								STR.s rd, addr32, ri	3	4[a]
0	CC		ZZZ			DD		10001100 (8c)								STR.s rd, addr32, ri	3	4[a]
0	AA		XXX			DD		10000001 (81)								STD.s rd, (addr16, ri)	2	5[a]
0	AA		YYY			DD		10010001 (81)								STD.s rd, (addr16), ri	2	5[a]

[a] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size

```
00:  8   01:  16   10:  32   11:  -
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
•	•	•	–	–	•

```
rd, C ← rd - <operand>
```

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	AA		SSS			DD		1	1	1	1	1	0	0		SUB.s rd, rs	1	1
1	BB		SSS			DD		1	1	0	0	0	1	1		SUB.s rd, rs	1	1
1	CC		SSS			DD		1	1	1	1	0	0	1		SUB.s rd, rs	1	1
1	AAAA				0	DD		1	1	1	0	1	0	0	1	SUB.s rd, #imm	2^a	2^b
1	AA		XXX			DD		1	1	1	1	0	1	0	1	SUB.s rd, addr16, ri	2	2^b
1	AA		ZZZ			DD		1	1	0	0	1	0	1	1	SUB.s rd, addr16, ri	2	2^b
1	AA		XXX			DD		1	1	1	1	1	1	0	1	SUB.s rd, addr32, ri	3	4^b
1	BB		XXX			DD		1	1	1	0	1	1	1	1	SUB.s rd, addr32, ri	3	4^b
1	CC		XXX			DD		1	1	1	1	1	1	1	1	SUB.s rd, addr32, ri	3	4^b
1	AA		YYY			DD		1	1	1	1	1	0	0	1	SUB.s rd, addr32, ri	3	4^b
1	BB		YYY			DD		1	1	0	1	0	1	1	1	SUB.s rd, addr32, ri	3	4^b
1	CC		YYY			DD		1	1	1	1	0	1	1	1	SUB.s rd, addr32, ri	3	4^b
1	AA		ZZZ			DD		1	1	0	1	1	0	1	1	SUB.s rd, addr32, ri	3	4^b
1	AA		XXX			DD		1	1	0	0	0	0	0	1	SUB.s rd, (addr16, ri)	2	5^b
1	AA		YYY			DD		1	1	1	1	0	0	0	1	SUB.s rd, (addr16, ri)	2	5^b

[a] add 1 word for 32 bit (s=2)
[b] add 1 cycle for 32 bit (s=2)

AA: Destination register rd

```
00:  A0   01:  A1   10:  A2   11:  A3
```

AAAA: Destination register rd

```
0000:  A0   0001:  A1   0010:  A2   0011:  A3
0100:  X0   0101:  X1   0110:  X2   0111:  X3
1000:  Y0   1001:  Y1   1010:  Y2   1011:  Y3
1100:  -    1101:  Z    1110:  SP   1111:  -
```

BB: Destination register rd

```
00:  X0   01:  X1   10:  X2   11:  X3
```

CC: Destination register rd

```
00:  Y0   01:  Y1   10:  Y2   11:  Y3
```

DD: Size s

```
00:  b 8   01:  w 16   10:  l 32   11:  -
```

SSS: Source register rs

```
000:  A0   001:  A1   010:  A2   011:  A3
100:  X0   101:  X1   110:  X2   111:  X3
```

XXX: Index register ri

```
000:  X0   001:  X1   010:  X2   011:  X3
100:  A0   101:  A1   110:  A2   111:  A3
```

YYY: Index register ri

```
000:  Y0   001:  Y1   010:  Y2   011:  Y3
100:  0    101:  Z    110:  SP   111:  PC
```

ZZZ: Index register ri

```
000:  0    001:  Z    010:  SP   011:  PC
100:  Y0   101:  Y1   110:  Y2   111:  Y3
```

Operation

N	Z	C	I	D	V
–	•	–	–	–	–

Z ← <operand> ∧ mask = 0

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1		AAA				RRRR					00011000 (18)					TBT rd, #imm	1	1
1		AAA				XXXX					00010010 (12)					TBT addr32, ri, #imm	3	5
1		BBB				RRRR					00011000 (58)					TBT rd, #imm	1	1
1		BBB				XXXX					00010010 (52)					TBT addr32, ri, #imm	3	5
1		CCC				RRRR					00011000 (d8)					TBT rd, #imm	1	1
1		CCC				XXXX					00010010 (d2)					TBT addr32, ri, #imm	3	5
1		DDD				RRRR					00011000 (b8)					TBT rd, #imm	1	1
1		DDD				XXXX					00010010 (92)					TBT addr32, ri, #imm	3	5

AAA: Bit number

000:	0	001:	7	010:	8	011:	15
100:	16	101:	23	110:	24	111:	31

BBB: Bit number

000:	2	001:	5	010:	10	011:	13
100:	18	101:	21	110:	26	111:	29

CCC: Bit number

000:	3	001:	4	010:	11	011:	12
100:	19	101:	20	110:	27	111:	28

DDD: Bit number

000:	6	001:	1	010:	14	011:	9
100:	22	101:	17	110:	30	111:	25

RRRR: Operand register rd

0000:	P	0001:	Z	0010:	SP	0011:	-
0100:	Y0	0101:	Y1	0110:	Y2	0111:	Y3
1000:	X0	1001:	X1	1010:	X2	1011:	X3
1100:	A0	1101:	A1	1110:	A2	1111:	A3

XXXX: Index register ri

0000:	A0	0001:	A1	0010:	A2	0011:	A3
0100:	X0	0101:	X1	0110:	X2	0111:	X3
1000:	Y0	1001:	Y1	1010:	Y2	1011:	Y3
1100:	0	1101:	Z	1110:	SP	1111:	-

Operation

N	Z	C	I	D	V
–	–	–	–	–	–

`<operand>` ← `<operand>` ⊕ `mask`

Encoding

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Syntax	Words	Cycles
1	AAA			RRRR				00111000 (38)								XBT rd, #imm	1	1
1	AAA			XXXX				00110010 (32)								XBT addr32, ri, #imm	3	5
1	BBB			RRRR				01111000 (78)								XBT rd, #imm	1	1
1	BBB			XXXX				01110010 (72)								XBT addr32, ri, #imm	3	5
1	CCC			RRRR				11111000 (f8)								XBT rd, #imm	1	1
1	CCC			XXXX				11110010 (f2)								XBT addr32, ri, #imm	3	5
1	DDD			RRRR				10011111 (9f)								XBT rd, #imm	1	1
1	DDD			XXXX				10110010 (b2)								XBT addr32, ri, #imm	3	5

AAA: Bit number

000:	0	001:	7	010:	8	011:	15
100:	16	101:	23	110:	24	111:	31

BBB: Bit number

000:	2	001:	5	010:	10	011:	13
100:	18	101:	21	110:	26	111:	29

CCC: Bit number

000:	3	001:	4	010:	11	011:	12
100:	19	101:	20	110:	27	111:	28

DDD: Bit number

000:	6	001:	1	010:	14	011:	9
100:	22	101:	17	110:	30	111:	25

RRRR: Operand register `rd`

0000:	P	0001:	Z	0010:	SP	0011:	-
0100:	Y0	0101:	Y1	0110:	Y2	0111:	Y3
1000:	X0	1001:	X1	1010:	X2	1011:	X3
1100:	A0	1101:	A1	1110:	A2	1111:	A3

XXXX: Index register `ri`

0000:	A0	0001:	A1	0010:	A2	0011:	A3
0100:	X0	0101:	X1	0110:	X2	0111:	X3
1000:	Y0	1001:	Y1	1010:	Y2	1011:	Y3
1100:	0	1101:	Z	1110:	SP	1111:	-