

## Podstawowe operacje2

### Instrukcje skoków bezwarunkowych i warunkowych

**JUMP** etykieta //skok bezwarunkowy

**CALL** etykieta //bezw warunkowe wywołanie podprogramu  
**IF NE JUMP** etykieta //skok warunkowy

**IF AC CALL** etykieta //warunkowe wywołanie podprogramu

Niektóre pozostałe instrukcje także mogą być wykonywane warunkowo, np. :

**IF EQ DM(I0,M0) = R2;**

**IF EQ R8 = R2;**

### Instrukcje skoków bezwarunkowych i warunkowych

#### - dozwolone warunki instrukcji warunkowych

Condition From	Description	True if...	Mnemonic
ALU	ALU = 0	AZ = 1	EQ
	ALU ≠ 0	AZ = 0	NE
	ALU > 0	footnote <sup>1</sup>	GT
	ALU < zero	footnote <sup>2</sup>	LT
	ALU ≥ 0	footnote <sup>3</sup>	GE
	ALU ≤ 0	footnote <sup>4</sup>	LE
	ALU carry	AC = 1	AC
	ALU not carry	AC = 0	NOT AC
	ALU overflow	AV = 1	AV
	ALU not overflow	AV = 0	NOT AV
Multiplier	Multiplier overflow	MV = 1	MV
	Multiplier not overflow	MV = 0	NOT MV
	Multiplier sign	MN = 1	MS
	Multiplier not sign	MN = 0	NOT MS

Condition From	Description	True if...	Mnemonic
Shifter	Shifter overflow	SV = 1	SV
	Shifter not overflow	SV = 0	NOT SV
	Shifter zero	SZ = 1	SZ
	Shifter not zero	SZ = 0	NOT SZ
Bit Test	Bit test flag true	BTF = 1	TF
	Bit test flag false	BTF = 0	NOT TF
Flag Input	Flag0 asserted	FI0 = 1	FLAG0_IN
	Flag0 not asserted	FI0 = 0	NOT FLAG0_IN
	Flag1 asserted	FI1 = 1	FLAG1_IN
	Flag1 not asserted	FI1 = 0	NOT FLAG1_IN
	Flag2 asserted	FI2 = 1	FLAG2_IN
	Flag2 not asserted	FI2 = 0	NOT FLAG2_IN
	Flag3 asserted	FI3 = 1	FLAG3_IN
	Flag3 not asserted	FI3 = 0	NOT FLAG3_IN
Mode	Bus master true		BM
	Bus master false		NOT BM

Condition From	Description	True if...	Mnemonic
Sequencer	Loop counter expired (Do)	CURLCNTR = 1	LCE
	Loop counter not expired (IF)	CURLCNTR ≠ 1	NOT ICE
	Always false (Do)	Always	FOREVER
	Always true (IF)	Always	TRUE

1 ALU greater than (GT) is true if:  $[\overline{AF} \text{ and } (AN \text{ xor } (AV \text{ and } \overline{ALUSAT})) \text{ or } (AF \text{ and } AN)] \text{ or } \overline{AZ} = 0$

2 ALU less than (LT) is true if:  $[\overline{AF} \text{ and } (AN \text{ xor } (AV \text{ and } \overline{ALUSAT})) \text{ or } (AF \text{ and } AN \text{ and } \overline{AZ})] = 1$

3 ALU greater equal (GE) is true if:  $[\overline{AF} \text{ and } (AN \text{ xor } (AV \text{ and } \overline{ALUSAT})) \text{ or } (AF \text{ and } AN \text{ and } \overline{AZ})] = 0$

4 ALU lesser or equal (LE) is true if:  $[\overline{AF} \text{ and } (AN \text{ xor } (AV \text{ and } \overline{ALUSAT})) \text{ or } (AF \text{ and } AN)] \text{ or } \overline{AZ} = 1$

## Skoki opóźnione - Delayed Branch

Skok nastąpi po wykonaniu jeszcze dwóch instrukcji (występujących za instrukcją skoku)

```
JUMP (db) etykieta //skok bezwarunkowy
```

```
CALL (db) etykieta //bezwarunkowe wywołanie podprogramuPrzykład:
```

```
    R0 = DM(I0,M0);
```

```
    JUMP (db) etykieta
```

```
    R1 = R2 + R3;
```

```
    R8 = 0;
```

```
    ... //ta instrukcja już się nie wykona !
```

```
    ...
```

```
    ...
```

```
Etykieta:
```