

The instruction set gives the programmer the ability to carry out all the functions necessary to program the system efficiently and may be divided into ten basic groups:

- Load/Store Instructions
- Arithmetic Instructions
- Logical Instructions
- Character Instructions
- Branch Instructions
- Shift Instructions
- Control Instructions
- Input/Output Instructions
- External Transfer Instructions
- Move Table Instructions

Within these groups efficiency is ensured by the possible use of up to eight different methods of forming one of the instruction's operands, the method to be used being chosen by the programmer with reference to the memory and timing requirements of any particular program.

Two formats for instruction layouts are used and where necessary two words are used to define an instruction.

INSTRUCTION FORMATS

Two instruction formats are possible and these are defined within the instruction by the most significant bit, bit 0, of the instruction word. Where instructions consist of two words the format bit is the most significant bit of the first word only.

Format 0 instructions are always short, that is one word. Format 1 instructions may be short or long, one or two words.

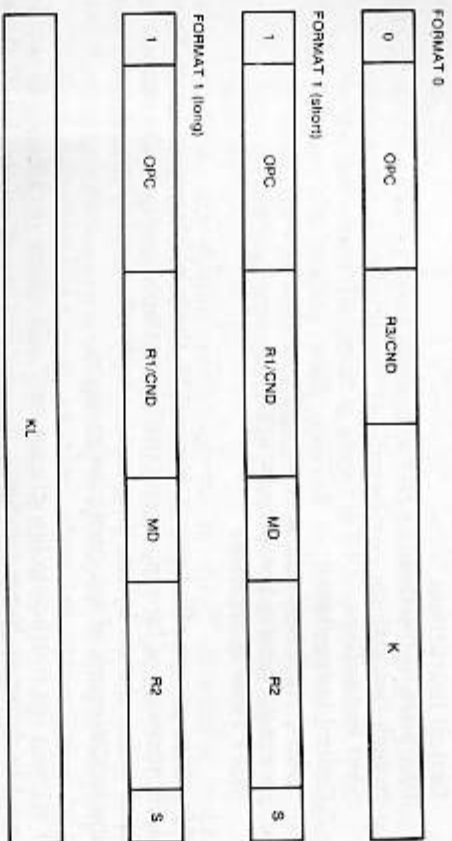


Figure 7.1 Layout of instruction formats.

OPC 4 bits, the pattern of which defines the instruction to be carried out.

R1 4 bits, specifies the working register to be used by the instruction, A0 - A15. It may contain one of the operands to be used and may also be used to hold the result of the instruction. In certain cases with R1 = 0 the addressed register, the P register, will not be used and in these cases R1 = 0 will qualify the operation code and define a different instruction than when R1 = 0.

R2 4 bits, specifies the second working register to be used by the instruction A1 - A15. It may contain the second operand or hold an address to be used in forming this operand. If R2 is made zero, no second working register is specified but this condition is used in deciding the method of forming the second operand.

R3 3 bits, specifies the working register to be used by the instruction A0 - A7. It may contain one of the operands to be used and may also be used to hold the result of the instruction. In certain cases with R3 = 0 the addressed register, the P register, will not be used and in these cases R3 = 0 will qualify the operation code and define a different instruction than when R3 = 0.

CND 3 bits, specifies the condition which must exist for a particular instruction to be carried out. Used to qualify conditional branch instructions and replaces R3 or the most significant 3 bits of R1.

MD 2 bits, specifies the mode of addressing to be used when forming the second operand of an instruction where this is applicable.

S 1 bit, applicable to certain instructions using memory. When present it specifies that the result of the instruction concerned is to be stored in the memory address specified by the instruction. When this bit is not present the result is placed into the working register specified by R1.

K 8 bits, these bits are used to specify the operand in format 0 instructions, and include short constant operands (K) and short displacements (m-for relative branch instructions). This field is also used to specify counts for shift instructions (n) and device addresses to I/O instructions (dev), in these cases a part of the field may be used to qualify the operation code.

KL 16 bits, this field is made up of the complete second word of a double length instruction and may specify a long constant (KL) or an address (m).

FORMING THE OPERAND

Many of the instructions may use various methods of forming one of the operands to be used. In all, eight methods of forming an operand are available governed by the values of the Format, Mode, and R2 fields of the instruction layout.

Figure 7.2 lists the eight methods of forming an operand and a brief description of each method is given following the figure.

Type	Format	Mode	R2	Reg/Reg.
T1	1	00	-	Reg/Reg.
T2	1	01	R2 = 0	Long Constant
T3	1	01	R2 ≠ 0	Address in Reg R2
T4	1	10	R2 = 0	Address in next word
T5	1	10	R2 ≠ 0	Indexed
T6	1	11	R2 = 0	Indirect
T7	1	-	R2 ≠ 0	Indexed Indirect
T8	0	-	-	Short Constant

Figure 7.2

- T1. **Register/Register - Format 1 (short)**
The operand is the value in the register specified by R2 of the instruction format.
- T2. **Long Constant - Format 1 (long)**
The operand is the value in the least significant word, all sixteen bits, of the double length instruction format.
- T3. **Address in Register - Format 1 (short)**
The operand is held in memory. The memory address of the operand is the value in the register specified by R2 of the instruction format.
- T4. **Address in Next Word - Format 1 (long)**
The operand is held in memory. The memory address of the operand is the value in the least significant word of the double length instruction.
- T5. **Indexed Address in Next Word - Format 1 (long)**
The operand is held in memory. The memory address of the operand is found by adding the value in the register specified by R2 of the instruction format to the value in the least significant word of the double length instruction.

T6. **Indirect Address in Next Word - Format 1 (long)**
The operand is held in memory. The memory address of the operand is also held in memory. This indirect address is the value in the least significant word of the double length instruction.

T7. **Indexed Indirect Address in Next Word - Format 1 (long)**
The operand is held in memory. The memory address of the operand is also held in memory. This indirect address is found by adding the value in the register specified by R2 of the instruction format to the value in the least significant word of the double length instruction.

T8. **Short Constant - Format 0**
The operand is the value in the least significant eight bits of the instruction format.

INSTRUCTION TIMING

The timing of the instructions depends on various factors: the type of instruction itself, the memory, the method of forming the operand and the number of memory cycles required.
The instruction set offers the possibility of very rapid execution times where single word register/register or short constant operations are employed whilst the more complex register/memory instructions save execution time when compared with the routines they may replace.
Execution time is also reduced in the case of conditional instructions by carrying out the conditional check immediately after accessing the instruction and then only continuing if the required conditions are satisfied.

TRAP ACTION

The use of any invalid instruction causes the activation of the Trap action which consists of the following basic actions:

- the CPU does not attempt to carry out the instruction
- information with reference to the instruction address and processor status is saved in the stack
- interrupts are inhibited
- a user mode flag is reset when working in user mode
- an indirect branch is made to address /7E for a trap routine.

no RIT, also automatically deactivated

THE INSTRUCTION SET

The instructions within the basic groups, together with their mnemonic, addressing type(s) and the execution time for the different types of memory are listed here:

Load/Store Instructions	Addressing types	Execution times in μ s	
		1.2 μ s memory	0.7 μ s memory
LD Load	T4 - T7	3.7 - 5.0 μ s	2.2 - 3.0 μ s
LDR Load Register	T1, T3	1.4 - 2.5 μ s	1.2 - 1.8 μ s
LDK Load Constant	T8, T2	1.3 - 2.5 μ s	0.9 - 1.5 μ s
ST Store	T4 - T7	3.8 - 5 μ s	2.4 - 3.3 μ s
STR Store Register	T3	2.8 μ s	2.1 μ s
ML Multiple Load	T4 - T7	2.8 - 4.1 +	2.6 - 3.5 +
MLR Multiple Load Register	T3	nx1.3 μ s	nx0.8 μ s
MLK Multiple Load Constant	T2	2.0 - 2.4 +	1.9 - 2.3 +
MS Multiple Store	T4 - T7	2.8 - 4.1 +	2.6 - 3.5 +
MSR Multiple Store Register	T3	nx1.3 μ s	nx0.8 μ s
EL Extended Load (MMU)	T4 - T7	2.5 - 3.1 +	2.3 - 2.9 +
ELR Extended Load Register (MMU)	T3	nx1.3 μ s	nx0.8 μ s
ES Extended Store (MMU)	T4 - T7	3 - 4.1 μ s	2.4 - 3.3 μ s
ESR Extended Store Register (MMU)	T3	2.5 μ s	2.1 μ s
Arithmetic Instructions			
AD Add	T4 - T7	3.8 - 6.3 μ s	2.2 - 3.9 μ s
ADR Add Register	T1, T3	1.4 - 3.8 μ s	1.2 - 2.6 μ s
ADK Add Constant	T8, T2	1.3 - 2.5 μ s	0.9 - 1.5 μ s
SU Subtract	T4 - T7	3.8 - 6.3 μ s	2.2 - 3.9 μ s
SUR Subtract Register	T1, T3	1.4 - 3.8 μ s	1.2 - 2.6 μ s
SUK Subtract Constant	T8, T2	1.3 - 2.5 μ s	0.9 - 1.5 μ s
MU Multiply	T4 - T7	9.7 - 11 μ s	8.6 - 9.5 μ s
MUR Multiply Register	T1, T3	7.8 - 8.5 μ s	7.6 - 8.9 μ s
MUK Multiply Constant	T2	8.5 μ s	7.9 μ s
DV Divide	T4 - T7	10 - 11.3 μ s	8.8 - 9.5 μ s
DVR Divide Register	T1, T3	7.8 - 8.8 μ s	7.6 - 8.9 μ s
DVK Divide Constant	T2	8.8 μ s	8.2 μ s

Load/Store Instructions	Addressing types	Execution times in μ s	
		1.2 μ s memory	0.7 μ s memory
DA Double Add	T4 - T7	5.6 - 6.9 μ s	3.9 - 4.8 μ s
DAR Double Add Register	T1, T3	3.1 - 4.5 μ s	3.0 - 3.6 μ s
DAK Double Add Constant	T2	4.4 μ s	3.2 μ s
DS Double Subtract	T4 - T7	5.6 - 6.9 μ s	3.9 - 4.8 μ s
DSR Double Subtract Register	T1, T3	3.1 - 4.5 μ s	3.0 - 3.6 μ s
DSK Double Subtract Constant	T2	4.4 μ s	3.2 μ s
C2 Two's Complement	T4 - T7	5.3 - 6.5 μ s	3.5 - 4.4 μ s
C2R Two's Complement Register	T3	4.0 μ s	3.1 μ s
IM Increment Memory	T4 - T7	5.0 - 6.3 μ s	3.0 - 4.0 μ s
IMR Increment Register	T3	3.8 μ s	2.6 μ s
NGR Negate Register	T1	2.0 μ s	1.9 μ s
CM Clear Memory	T4 - T7	3.8 - 5.0 μ s	2.4 - 3.3 μ s
CMR Clear Register	T3	2.8 μ s	2.1 μ s
CW Compare Word	T4 - T7	3.8 - 5.0 μ s	2.2 - 3.0 μ s
CWR Compare Word Register	T1, T3	1.4 - 2.5 μ s	1.2 - 1.8 μ s
CWK Compare Word Constant	T2	2.5 μ s	1.5 μ s
FPL Integer to Floating Point	T1		3.7 μ s
FFX Floating Point to Integer	T1		5.1 μ s
FAD Floating Point Addition	T4 - T7		6.2 - 9.6 μ s
FADR Floating Point Addition/Register	T3		5.9 - 8.4 μ s
FSU Floating Point Subtract	T4 - T7		6.2 - 9.6 μ s
FSUR Floating Point Subtract/Register	T3		5.9 - 8.4 μ s
FMU Floating Point Multiply	T4 - T7		8.8 - 12.2 μ s
FMUR Floating Point Multiply/Register	T3		8.4 - 11.0 μ s