8-bit CPU

Alexay Mehra

1 Overview

The Instruction Set Architecture (ISA) of the 8-bit CPU is defined as follows:

Memory Address Space The memory address space is defined by an 8-bit architecture, corresponding to 2^8 (256) unique locations. Each location contains one word (8 bits), representing either an instruction or data. Addresses are numbered from 0 (x00) to 255 (xFF).

Bit Numbering Bits are numbered from right to left, starting with bit 0. In an 8-bit word, the rightmost bit (the least significant bit) is bit 0 and the leftmost bit (the most significant bit) is bit 7.

Instruction Register (IR) An 8-bit register that contains the current instruction to be processed. Instructions are exactly 8 bits wide. Bits [7:5] specify the opcode (the operation to be performed), while bits [4:0] provide other information, such as register identifiers or immediate values (offsets). The specific bit-level formats are detailed in Figure 1.

Program Counter (PC) An 8-bit register that contains the address of the next instruction to be fetched and processed. Under normal execution, the PC increments by 1 after each instruction fetch.

General Purpose Registers The CPU contains two 8-bit general-purpose registers, identified as A and B. In the instruction format, these are represented by the DR (Destination Register), SR (Source Register), or BaseR fields. A bit value of 0 identifies A, and a bit value of 1 identifies B.

Zero Flag Register (ZF Register) A 1-bit register that indicates if the result of the previous arithmetic or logical operation resulted in 0. Only ALU (Arithmetic Logic Unit) instructions (ADD, AND, and NOT) set this condition. All other instructions leave this condition unchanged.

ALU Instructions These perform arithmetic or logic operations on data stored in registers:

- **ADD**: Performs 2's complement addition.
- **AND**: Performs a bitwise logical AND.
- **NOT**: Performs a bitwise logical complement (flips the bits).

Data Movement Instructions These move data between memory and registers using a 4-bit (positive-only) PC-relative offset:

- **LOAD**: Copies data from memory into a Destination Register (DR).
- **STORE**: Copies data from a Source Register (SR) into a memory location.

Control Instructions These alter the sequence of execution by loading a new address into the Program Counter that is specified by BaseR and a 4-bit (positive-only) offset:

- **JUMP**: An unconditional branch to the specified address.
- **JUMPz**: A conditional branch that occurs only if the zero flag is set.

	7	6	5	4	3	2	1	0
ADD	0	0	0	DR	SR1	SR2	0	0
AND	0	0	1	DR	SR1	SR2	0	0
NOT	0	1	0	DR	SR	0	0	0
LOAD	0	1	1	DR	PCoffset4			
STORE	1	0	0	SR	PCoffset4			
JUMP	1	0	1	BaseR	offset4			
JUMPz	1	1	0	BaseR	offset4			
HALT	1	1	1	1	1	1	1	1

Figure 1: Format of the entire 8-bit CPU instruction set.

2 Instruction Set Specifications

ADD

Assembler Format

ADD DR, SR1, SR2

Encoding

7	6	5	4	3	2	1	0
0	0	0	DR	SR1	SR2	0	0

Operation

DR = SR1 + SR2

Examples

ADD A, B, A ; A ← B + A (00001000)
ADD B, A, A ; B ← A + A (00010000)

AND

Assembler Format

AND DR, SR1, SR2

Encoding

7	6	5	4	3	2	1	0
0	0	1	DR	SR1	SR2	0	0

Operation

DR = SR1 AND SR2

Examples

AND A, B, A ; A ← B AND A (00101000)
AND B, A, B ; B ← A AND B (00110100)

NOT

Assembler Format

NOT DR, SR

Encoding

7	6	5	4	3	2	1	0
0	1	0	DR	SR	0	0	0

Operation

DR = NOT SR

Examples

NOT A, A ; A ← NOT A (01000000)
NOT B, A ; B ← NOT A (01010000)

LOAD

Assembler Format

LOAD DR, PCoffset4

Encoding

7	6	5	4	3	2	1	0
0	1	1	DR	PCoffset4			

Operation

DR = mem[PC + ZEXT(PCoffset4)]
Note: The PC is incremented during the instruction fetch phase, before the evaluation of the effective address.

Examples

LOAD A, 1111 ; A ← mem[PC + 15] (01101111)
LOAD B, 1010 ; B ← mem[PC + 10] (01111010)

STORE

Assembler Format

STORE SR, PCoffset4

Encoding

7	6	5	4	3	2	1	0
1	0	0	SR	PCoffset4			

Operation

mem[PC + ZEXT(PCoffset4)] = SR
Note: The PC is incremented during the instruction fetch phase,
before the evaluation of the effective address.

Examples

STORE A, 1111 ; mem[PC + 15] $\leftarrow$ A (10001111)
STORE B, 1010 ; mem[PC + 10] $\leftarrow$ B (10011010)

JUMP

Assembler Format

JUMP BaseR, offset4

Encoding

7	6	5	4	3	2	1	0
1	0	1	BaseR	offset4			

Operation

PC = BaseR + ZEXT(offset4)

Examples

JUMP A, 1111 ; PC $\leftarrow$ A + 15 (10101111)
JUMP B, 1010 ; PC $\leftarrow$ B + 10 (10111010)

JUMPz

Assembler Format

JUMPz BaseR, offset4

Encoding

7	6	5	4	3	2	1	0
1	1	0	BaseR	offset4			

Operation

if (zf) PC = BaseR + ZEXT(offset4)

Examples

JUMPz A, 1111 ; (if zf = 1) PC $\leftarrow$ A + 15 (11001111)
JUMPz B, 1010 ; (if zf = 1) PC $\leftarrow$ B + 10 (11011010)

HALT

Assembler Format

HALT

Encoding

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	1

Operation

Stops execution and holds the processor in an idle state until reset.

Examples

HALT ; (11111111)