Specifications of the processor

Processor follows 16-bit instructions and addressing mechanisms. Assembly language instruction format is:

Opcode<operand1><operand2><operand3>

(Where, operand1 is the destination register)

Processor can execute following instructions -

* LOAD - loads data from memory into registers LOAD R1, #4 or LOAD R1,
* STORE - stores data from register into memory location STORE R1, LOC
* ADD - compute addition of operand2 and operand3 and stores the result at

operand3

* SUBTRACT - compute subtraction of operand2 and operand3 and stores the

result at operand3

* AND - compute bitwise AND of operand2 and operand3 and stores the

result at operand3

* OR - compute bitwise OR of operand2 and operand3 and stores the result at

operand3

* XOR - compute bitwise XOR of operand2 and operand3 and stores the

result at operand3

**INSTRUCTION ENCODING**

Assembly instructions are converted into machine instructions by following a certain encoding. The encoding is as follows –

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| <1-Bit> | <4-Bits> | <1-Bit> | <3-Bits> | <4-Bits> | <3-Bits> |
| T | Opcode | I | Operand-1 | Operand-2 | Operand-3 |

First 1 bit (T-bit) represents the type of instruction; whether the instruction is arithmetic or logical instruction (Add, Sub, etc.) or data transfer instruction (Load, Store). If T is 1 then instruction is data transfer instruction else arithmetic or logical instruction.

Next 4 bits of instruction represent the instruction Opcode. Which is used to identify what operation to perform.

* The 0000 bit sequence represents ADD instruction.
* The 0001 bit sequence represents SUB instruction.
* The 0010 bit sequence represents AND instruction.
* The 0011 bit sequence represents OR instruction.
* The 0100 bit sequence represents XOR instruction.
* The 0101 bit sequence represents LOAD instruction.
* The 0110 bit sequence represents STORE instruction.

Data transfer instructions, load and store will follow the follwoing format

<Opcode><Register><Memory Location>

I-bit will always be 0 for these instructions as operand2 should be an address of memory location.

Operand2 will always be a memory address for data transfer instructions.

For other instructions it can be either a constant or general purpose register.

Since these instructions only have two operands, so operand3 (last 3-bits) will always be null represented as 000.

Next 1 bit (I bit) is used to represent whether operand2 is an immediate value or not. If I-bit is set to 1 which means that operand2 should be treated as a constant value. The constant value is represented in 4-bit 2’s complement system.

Next 3-bits represents operand1 which will always be a register.

* 000 - R0
* 001 - R1
* 010 - R2
* 011 - R3
* 100 - R4
* 101 - R5
* 110 - R6
* 111 - R7

Next 4-bits represents operand2 which can be either a register or a constant value depending on the I-bit. If operand2 is a register then -

* 0000 - R0
* 0001 - R1
* 0010 - R2
* 0011 - R3
* 0100 - R4
* 0101 - R5
* 0110 - R6
* ○ 0111 - R7

Next 3-bits represents operand3 which will also be a register always except for Load and store instructions.

* 000 - R0
* 001 - R1
* 010 - R2
* 011 - R3
* 100 - R4
* 101 - R5
* 110 - R6
* 111 - R7

If the instruction type is either Load or Store then the last 3-bits will always be 0 as Load and store instructions only require 2 operands