## **Mostly Optical Simple Instruction Set (MOSIS)**

The following is a very simple 16-bit instruction set that we designed for the exercises. While it’s not that complicated, it strongly resembles other real world architectures. It is designed to be easily disassembled by hand, since most of the fields are one nybble (four bits, one hexadecimal digit) wide.

***MOSIS Notes:***

Instruction Encoding: All instructions are 16-bits. The first 4 bits represent the opcode, and the remaining 12 bits are used in an instruction-dependent way. Instructions have between 0 and 3 parameters. Most of the instruction names are fixed, except the conditional jump instruction (J*cc*). The conditional jump instruction’s name can be rendered differently depending on the condition (e.g. it becomes JLT when checking for “less than” or JNE when checking for “not equal).

Registers: 16, R0 – R15. There is also an invisible “flags” register for comparison operations, and an invisible instruction pointer. Registers are 16-bits wide.

Addressing: Since instructions are 16-bits, to make things easy addresses are 16-bit aligned. That means that the parameters in the jump and call instructions are a number of instructions, not a number of bytes. It (should be!) impossible to make the CPU execute starting on an odd byte.

MOV – copy the contents of one register to another

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Unused (4 bits) | Destination Register (4 bits) | Source Register (4 bits) |
| 0 0 0 0 | 0 0 0 0 | Y Y Y Y | X X X X |

Set register Y equal to register X.

Examples:

0012 MOV R1, R2 Set R1 equal to R2.

0021 MOV R2, R1 Set R2 equal to R1.

MOVI – move a number encoded in the instruction into a register

|  |  |  |
| --- | --- | --- |
| Opcode (4 bits) | Destination Register (4 bits) | Immediate Value (8 bits) |
| 0 0 0 1 | Y Y Y Y | I I I I I I I I |

Set register Y equal to the value of I.

Examples:

1122 MOVI R1, 0x22 Set R1 equal to 0x22 (34).

1B01 MOVI R11, 1 Set R11 equal to 1.

***ADD – add two registers and store the result in another register***

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Sum Register (4 bits) | Addend 1 (4 bits) | Addend 2 (4 bits) |
| 0 0 1 0 | Z Z Z Z | X X X X | Y Y Y Y |

Add register X and register Y and store the result in register Z.

Examples:

2312 ADD R3, R1, R2 Set R3 = R1 + R2.

2A34 ADD R10, R3, R4 Set R10 = R3 + R4.

***SUB – subtract two registers and store the result in another register***

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Difference (4 bits) | Subtrahend (4 bits) | Minuend (4 bits) |
| 0 0 1 1 | Z Z Z Z | X X X X | Y Y Y Y |

Subtract register Y (minuend) from register X (subtrahend) and store the difference in register Z.

Examples:

3A98 SUB R10, R9, R8 Set R10 = R9 - R8.

3FED SUB R15, R14, R13 Set R15 = R14 - R13.

***CMP – compare two registers (subtract) and set the flags register (used by conditional jump)***

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Unused (4 bits) | Subtrahend (4 bits) | Minuend (4 bits) |
| 0 1 0 0 | 0 0 0 0 | X X X X | Y Y Y Y |

Calculate register X – register Y, but don’t save the result. Set an invisible condition register to be used later by the conditional jump instruction.

Examples:

4012 CMP R1, R2 Calculate R1-R2.

40BC CMP R11, R12 Calculate R11-R12.

***Jcc – jump if a certain condition is true (based on the result of a compare instruction)***

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Condition Code (4 bits) | Sign Bit | Instruction Offset (7 bits) |
| 0 1 0 1 | 0 0 0 0: Equal (JE)  0 0 0 1: Less than (JLT)  0 0 1 0: Greater than (JGT)  0 0 1 1: Less than or equal (JLTE)  0 1 0 0: Greater than or equal (JGTE)  0 1 0 1: Not equal (JNE) | 0: +  1: - | O O O O O O O |

If condition is true, jump to the offset (O is an instruction count, positive or negative based on the sign bit), otherwise execute the next instruction. Offset is from the *beginning* of the J*cc* instruction.

Examples:

4012 CMP R1, R2

5085 JEQ -5 If R1 = R2, then jump 5 instructions backward.

40AB CMP R10, R11

542A JGTE 42 If R10 >= R11, then jump 42 instructions forward.

***JMP – jump to a fixed address***

|  |  |
| --- | --- |
| Opcode (4 bits) | Address (12 bits) |
| 0 1 1 0 | A A A A A A A A A A A A |

Jump to an absolute fixed address (not an offset).

Examples:

61A2 JMP 0x1A2 Jump to program address 0x1A2

6EAB JMP 0xEAB Jump to program address 0xEAB

***CALL – call a function***

|  |  |
| --- | --- |
| Opcode (4 bits) | Address (12 bits) |
| 0 1 1 1 | A A A A A A A A A A A A |

Like the JMP instruction, except push the return address (address of the instruction immediately following this one) onto the call stack. This is done so that the RET instruction at the end of the function will jump back to the next instruction.

Examples:

7FCE CALL 0xFCE Call function 0xFCE

7123 CALL 0x123 Call function 0x123

***RET – return from a function***

|  |  |
| --- | --- |
| Opcode (4 bits) | Unused (12 bits) |
| 1 0 0 0 | 0 0 0 0 0 0 0 0 0 0 0 0 |

Pop the last return address off of the call stack and jump there.

Example:

8000 RET Return

***NOP – no operation***

|  |  |
| --- | --- |
| Opcode (4 bits) | Unused (12 bits) |
| 1 0 0 1 | 0 0 0 0 0 0 0 0 0 0 0 0 |

Do nothing.

Example:

9000 NOP No operation

***IN – read from external memory / device***

|  |  |  |
| --- | --- | --- |
| Opcode (4 bits) | Unused (8 bits) | Input Register (4 bits) |
| 1 0 1 0 | 0 0 0 0 0 0 0 0 | X X X X |

Read from memory or input device and store in register X.

Example:

A004 IN R4 Read from input to R4

A00C IN R12 Read from input to R12

***OUT – write to external memory / device***

|  |  |  |
| --- | --- | --- |
| Opcode (4 bits) | Unused (8 bits) | Input Register (4 bits) |
| 1 0 1 1 | 0 0 0 0 0 0 0 0 | X X X X |

Write register contents out to memory or device.

Example:

B005 OUT R5 Output R5

B00D OUT R13 Output R13

***MUL – multiply two registers and store the result in another register***

|  |  |  |  |
| --- | --- | --- | --- |
| Opcode (4 bits) | Product (4 bits) | Multiplicand (4 bits) | Multiplier (4 bits) |
| 1 1 0 0 | Z Z Z Z | X X X X | Y Y Y Y |

Multiply register X times register Y and store the product in register Z.

Examples:

CA98 MUL R10, R9, R8 Set R10 = R9 \* R8.

CFED MUL R15, R14, R13 Set R15 = R14 \* R13.