

CPU ISA Green Card

Registers

| Code | Name | Notes |
|------|------|-----------------|
| 00 | A | General purpose |
| 01 | B | General purpose |
| 10 | C | General purpose |
| 11 | IX | Index register |

Memory

- **IMEM:** 16 instructions (PC = 4 bits)
- **DMEM:** 8 bytes, byte-addressable (3-bit address)

Instruction Format

| | | | |
|----------------|-------------------------------|------------|------------------------------|
| Opcode [15:12] | Clarifier/AddrMode/R2 [11:10] | R(1) [9:8] | Imm8 / Addr / ShiftAmt [7:0] |
|----------------|-------------------------------|------------|------------------------------|

Addressing Modes

| Code | Meaning |
|------|------------------------------|
| 00 | Immediate (Imm8) |
| 01 | Direct (DMEM[addr3]) |
| 10 | Indirect (DMEM[DMEM[addr3]]) |
| 11 | Indexed (DMEM[IX+addr3]) |

Flags

| Flag | Meaning |
|------|---------------------|
| Z | Zero |
| C | Carry |
| N | Negative (sign bit) |
| O | Overflow |

Control Signals

| Singal | Usage |
|-------------|--|
| REG_WLINE_1 | 0x: ALU or Imm8, 1x: DMEM or IN |
| REG_WLINE_0 | x0: ALU or DMEM, x1: Imm8 or IN |
| REG_W_EN | Allow write to Main Regs |
| REG_W_ADD_1 | MReg write address MSB |
| REG_W_ADD_0 | MReg write address LSB |
| REG_R_ADD_4 | Output C = Sig ? Ones : Xeros |
| REG_R_ADD_3 | Read Address of Out B |
| REG_R_ADD_2 | Read Address of Out B |
| REG_R_ADD_1 | Read Address of Out A |
| REG_R_ADD_0 | Read Address of Out A |
| ADD/SUB | Select Add or Sub in Adder |
| ALU_S_1 | Select Out between ALU Ops |
| ALU_S_0 | Select Out between ALU Ops |
| FLAG_W | Flag Reg write enable |
| ALU_SL_1 | Select Y input of ALU |
| ALU_SL_0 | Select Y Input of ALU |
| DMEM_W_EN | DMEM Write Enable |
| DMEM_S_ADD | Select between ALU and Imm for Address |
| PC_LD_EN | Allows for a new address to be loaded |
| PC_EN | Allows for PC to increment |
| OUT_EN | Enable LCD Output |
| X_SEL | Used indexed DMEM out for ALU Y |
| I_SEL | Read indirectly from DMEM |

Instruction Set

| Opcode | Mode | Mnemonic | Description | Instruction Format |
|--------|------|----------|-----------------------------------|--------------------|
| 0000 | 00 | NOOP | Do nothing | NOOP |
| 0000 | 01 | IN | $R \leftarrow \text{Input}$ | IN R |
| 0000 | 10 | OUT | Output R (ASCII) | OUT R |
| 0000 | 11 | END | Terminate program | END |
| 0001 | 00 | LDM | Load operand into R | LDM R, imm8 |
| 0001 | 01 | LDD | Load operand into R | LDD R, addr3 |
| 0001 | 10 | LDI | Load operand into R | LDI R, addr3 |
| 0001 | 11 | LDX | Load operand into R | LDX R, addr3 |
| 0010 | 00 | SWAP | Swap registers | SWAP R1, R2 |
| 0010 | 01 | STO | Store R \rightarrow operand | STO R, imm8 |
| 0010 | 10 | STI | Store R \rightarrow operand | STI R, addr3 |
| 0010 | 11 | STX | Store R \rightarrow operand | STX R, addr3 |
| 0011 | 00 | ADDM | $R \leftarrow R + \text{operand}$ | ADDM R, imm8 |
| 0011 | 01 | ADDD | $R \leftarrow R + \text{operand}$ | ADDD R, addr3 |
| 0011 | 10 | ADDI | $R \leftarrow R + \text{operand}$ | ADDI R, addr3 |
| 0011 | 11 | ADDX | $R \leftarrow R + \text{operand}$ | ADDX R, addr3 |
| 0100 | 00 | SUBM | $R \leftarrow R - \text{operand}$ | SUBM R, imm8 |
| 0100 | 01 | SUBD | $R \leftarrow R - \text{operand}$ | SUBD R, addr3 |
| 0100 | 10 | SUBI | $R \leftarrow R - \text{operand}$ | SUBI R, addr3 |
| 0100 | 11 | SUBX | $R \leftarrow R - \text{operand}$ | SUBX R, addr3 |
| 0101 | 00 | INC | $R \leftarrow R + 1$ | INC R |
| 0101 | 01 | DEC | $R \leftarrow R - 1$ | DEC R |
| 0101 | 10 | LSL | Logical shift left | LSL R, shiftamt3 |
| 0101 | 11 | LSR | Logical shift right | LSR R, shiftamt3 |
| 0110 | – | ADDR | $R1 \leftarrow R1 + R2$ | ADDR R1, R2 |
| 0111 | – | SUBR | $R1 \leftarrow R1 - R2$ | SUBR R1, R2 |
| 1000 | – | MOV | $R1 \leftarrow R2$ | MOV R1, R2 |
| 1001 | 00 | CMP | Compare, set flags | CMP R, imm8 |
| 1001 | 01 | CMD | Compare, set flags | CMD R, addr3 |
| 1001 | 10 | CMI | Compare, set flags | CMI R, addr3 |
| 1001 | 11 | CMX | Compare, set flags | CMX R, addr3 |
| 1010 | – | SWAPR | Swap registers | SWAPR R1, R2 |
| 1010 | 01 | JMP | $PC \leftarrow \text{addr4}$ | JMP addr4 |
| 1011 | 00 | JPN | Jump if Z=0 | JPN addr4 |
| 1011 | 01 | JPE | Jump if Z=1 | JPE addr4 |
| 1011 | 10 | JGT | Jump if Z=0 and N=O | JGT addr4 |
| 1011 | 11 | JGE | Jump if N=O | JGE addr4 |
| 1100 | 00 | ANDM | Bitwise AND | ANDM R, imm8 |
| 1100 | 01 | ANDD | Bitwise AND | ANDD R, addr3 |
| 1100 | 10 | ANDI | Bitwise AND | ANDI R, addr3 |
| 1100 | 11 | ANDX | Bitwise AND | ANDX R, addr3 |
| 1101 | 00 | ORM | Bitwise OR | ORM R, imm8 |
| 1101 | 01 | ORD | Bitwise OR | ORD R, addr3 |
| 1101 | 10 | ORI | Bitwise OR | ORI R, addr3 |
| 1101 | 11 | ORX | Bitwise OR | ORX R, addr3 |
| 1110 | 00 | XORM | Bitwise XOR | XORM R, imm8 |
| 1110 | 01 | XORD | Bitwise XOR | XORD R, addr3 |
| 1110 | 10 | XORI | Bitwise XOR | XORI R, addr3 |
| 1110 | 11 | XORX | Bitwise XOR | XORX R, addr3 |
| 1111 | 00 | NOT | $R \leftarrow \sim R$ | NOT R |
| 1111 | 01 | ASR | Arithmetic shift right | ASR R, shiftamt3 |
| 1111 | 10 | CSL | Circular shift left | CSL R, shiftamt3 |
| 1111 | 11 | CSR | Circular shift right | CSR R, shiftamt3 |