# SIA32

## Instruction Formats

3 Register (3R) - 11

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Immediate (8) | Rs1(5) | Rs2(5) | Function(4) | Rd (5) | Opcode(5) |

2 Register (2R) - 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Immediate (13) | Rs(5) | Function(4) | Rd (5) | Opcode(5) |

Dest Only (1R) - 01

|  |  |  |  |
| --- | --- | --- | --- |
| Immediate (18) | Function(4) | Rd (5) | Opcode(5) |

No Register (0R) - 00

|  |  |
| --- | --- |
| Immediate (27) | Opcode(5) |

A SIA32 opcode is made up of two parts – the operation and the instruction format. The opcode is made by combining the instruction code and then the instruction format. For example – a 3R math operation is opcode 00011.

Rs1 and Rs2 are the two source registers; Rd is the destination register.

## Registers

There are 32 general purpose registers (R0 – R31). R0 is hard-coded to 0; writing to it leaves it unchanged (is a NO OP).

There are 2 special purpose registers: Stack pointer (SP), Program Counter (PC). These are not directly readable or writable but are changed by instructions like branch, call, return, push and pop.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 2R (11) | 3R (10) | Dest Only (01) | No R (00) |
| Math (000) | Rd 🡨 Rd MOP Rs | Rd 🡨 Rs1 MOP Rs2 | COPY: Rd🡨 imm | HALT |
| Branch (001) | pc 🡨 Rs BOP Rd? pc + imm : pc | pc 🡨 Rs1 BOP Rs2 ? pc + imm : pc | JUMP: pc 🡨 pc + imm | JUMP: pc 🡨 imm |
| Call (010) | pc 🡨 Rs BOP Rd? push pc; pc + imm : pc | pc 🡨 Rs1 BOP Rs2 ? push pc; Rd + imm : pc | push pc; pc 🡨 Rd + imm | push pc; pc 🡨 imm |
| Push (011) | mem[--sp] 🡨 Rd MOP Rs | mem[--sp] 🡨 Rs1 MOP Rs2 | mem[--sp] 🡨 Rd MOP imm | UNUSED |
| Load (100) | Rd 🡨 mem[Rs + imm] | Rd 🡨 mem [Rs1+ Rs2] | Rd 🡨 mem [Rd + imm] | RETURN (pc 🡨 pop) |
| Store (101) | mem[Rd + imm] 🡨 Rs | Mem[Rd + Rs1] 🡨 Rs2 | Mem[Rd] 🡨 imm | UNUSED |
| Pop/interrupt (110) | PEEK: Rd 🡨 mem[sp – (Rs + imm)] | PEEK: Rd 🡨 mem [sp – (Rs1+ Rs2)] | POP: Rd 🡨 mem[sp++] | INTERRUPT: Push pc; pc 🡨 intvec[imm] |

## Instruction Definition Matrix

imm = immediate value

mem = main memory

|  |  |
| --- | --- |
| MOP (math op) | |
| Bit Pattern | Meaning |
| 1000 | and |
| 1001 | or |
| 1010 | xor |
| 1011 | not (negate op1; ignore op2) |
| 1100 | left shift (“op1” is the value to shift, “op2” is the amount to shift; ignore all but the lowest 5 bits) |
| 1101 | right shift (“op1” is the value to shift, “op2” is the amount to shift; ignore all but the lowest 5 bits) |
| 1110 | Add |
| 1111 | Subtract |
| 0111 | Multiply |
| Other values are not valid | |

|  |  |
| --- | --- |
| BOP (boolean op) | |
| Bit Pattern | Meaning |
| 0000 | Equals (eq) |
| 0001 | Not Equal (neq) |
| 0010 | Less than (lt) |
| 0011 | Greater than or equal (ge) |
| 0100 | Greater than (gt) |
| 0101 | Less than or equal (le) |
| Other values are not valid | |