# CJG_RISC721 Instruction Set Architecture

## Register-Register (Load/Store) – 32-bits

- **General Specifications:**
  - One physical addressing mode
  - 32 general purpose registers
  - $2^{16}$ = 64KWord memory space for both DM and PM
  - Input/Output Peripherals (I/O-Ps) are memory mapped in the memory range 0xFFE0 to 0xFFFF;
  - Hardware Stack: 32-bits, 16 registers deep
  - Status Bits register (SR) is a 32-bit register located at R0:

| Significance | Int. En | <* | >=* | Zero | oVerflow | Negative | Carry |
|---|---|---|---|---|---|---|---|
| Field Size | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Index in R0 | 13–6 | 5 | 4 | 3 | 2 | 1 | 0 |

* Only set from the CMP operation

** Interrupt disable flag (one for each interrupt)

  - Interrupts served by a priority encoder (lower is first)
    - Will complete instructions in pipeline first and stall until ready to serve the ISR. R0 – R23 copied to shadow register file and restored on RETI.
    - Interrupt Vector Table from 0xFFF0 to 0xFFFE in the PM
  - Interrupt Enables

| ISR_7 | ISR_6 | ISR_5 | ISR_4 | ISR_3 | ISR_2 | ISR_1 | ISR_0 |
|---|---|---|---|---|---|---|---|
| - | - | Counter | Timer | Ext_Interrupt_Bus[3:0] | | | |
| 1 << 13 | 1 << 12 | 1 << 11 | 1 << 10 | 1 << 9 | 1 << 8 | 1 << 7 | 1 << 6 |

  - Memory Mapped I/O
    - 0x3FFF → Binary LED output
    - 0x3FFE → Switch Input
    - 0x3FFD → Timer module
    - 0x3FFC → Counter
    - 0x3FFB → 7-segment Display

- **Load/Store Instructions:**

Instruction Word:

| Significance | OpCode | Ri | Rj | Control | Address |
|---|---|---|---|---|---|
| Field Size | 5 | 5 | 5 | 1 | 16 |

  - Control for addressing mode:

| Rj | Control | Description |
|---|---|---|
| Not 0 | 0 | Indexed: Address is the Rj register value + Address field value |
| Not 0 | 1 | Register Direct: Address is the value of the Rj register |
| 0 | 0 | PC Relative: Address is the PC value + Address field |
| 0 | 1 | Absolute: Address is the value of the Address field |

  - LD = Load: load from memory location or input peripheral into Ri
  - ST = Store: store from Ri into memory location or output peripheral

**Data Transfer Instructions**

Instruction Word:

| Significance | OpCode | Ri | Rj | Control | Constant |
|---|---|---|---|---|---|
| Field Size | 5 | 5 | 5 | 1 | 16 |

- If control is 1, Rj is not used, the constant value is
  - CPY = Copy: copy from register Rj into register Ri
  - PUSH = Push Rj value onto top of stack
  - POP = Pop top of stack value into Ri

**Flow Control Instructions:**

Instruction Word:

| Significance | OpCode | Ri | C | N | V | Z | Unused | Control | Address |
|---|---|---|---|---|---|---|---|---|---|
| Field Size | 5 | 5 | 1 | 1 | 1 | 1 | 1 | 1 | 16 |

- Control for addressing mode:

| Ri | Control | Description |
|---|---|---|
| Not 0 | 0 | Indexed: Address is the Ri register value + Address field value |
| Not 0 | 1 | Register Direct: Address is the value of the Ri register |
| 0 | 0 | PC Relative: Address is the PC value + Address field |
| 0 | 1 | Absolute: Address is the value of the Address field |

  - Jump Condition table:

| C | N | V | Z | Mnemonic | Description |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | JMP / JU | Jump Unconditional |
| 1 | 0 | 0 | 0 | JC | Jump if C = 1 |
| 0 | 1 | 0 | 0 | JN | Jump if N = 1 |
| 0 | 0 | 1 | 0 | JV | Jump if V = 1 |
| 0 | 0 | 0 | 1 | JZ / JEQ | Jump if Z = 1 |
| 0 | 1 | 1 | 1 | JNC | Jump if C = 0 |
| 1 | 0 | 1 | 1 | JNN | Jump if N = 0 |
| 1 | 1 | 0 | 1 | JNV | Jump if V = 0 |
| 1 | 1 | 1 | 0 | JNZ / JNE | Jump if Z = 0 |
| 0 | 1 | 1 | 0 | JGE | Jump if greater or equal |
| 1 | 0 | 0 | 1 | JL | Jump if less than |

  - Jump: PC = Address (if condition is met)
  - CALL = Subroutine Call (push PC then SR)
    - Call to address field
  - RET = Return from subroutine (pop SR then pop and load PC)
    - If Control is 1, then RETI (same as RET but restores RF too)

- **Manipulation (ALU) Instructions:**

  Instruction Word:

| Significance | OpCode | Ri | Rj | Rk / Constant[1] | Control |
|---|---|---|---|---|---|
| Field Size | 5 | 5 | 5 | 5 / 16 | 1 |

  [1] For any instructions that use constant values, the constant value is taken as the [16:1]
    - If control is 1, Rk is not used, the constant value is used
  - ADD → Ri = Rj + Rk ; signed addition, 2's-Complement
  - SUB → Ri = Rj – Rk ; signed sub, 2's-Complement
  - CMP → Rj – Rk ; Set status bits but not result
  - NOT → Ri = NOT Rj ; Logical NOT
  - AND → Ri = Rj AND Rk ; Logical AND
  - BIC → Ri = Rj & ~Rk ; Bit Clear
  - OR → Ri = Rj OR Rk ; Logical OR
  - XOR → Ri = Rj XOR Rk ; Logical XOR

- **Rotate/Shift (Also ALU)**:

  Instruction Word

| Significance | OpCode | Ri | Rj | Rk / Constant[1] | Unused | Control | Constant_Control |
|---|---|---|---|---|---|---|---|
| Field Size | 5 | 5 | 5 | 5 / 6 | 8/7 | 3 | 1 |

  Control Table:

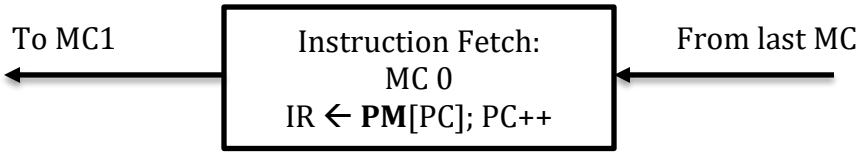
| Control | | | Mnemonic | Description |
|---|---|---|---|---|
| 0 | 0 | 0 | SRL | Shift right logical |
| 0 | 0 | 1 | SLL | Shift left logical |
| 0 | 1 | 0 | SRA | Shift right arithmetic |
| 1 | 0 | 0 | RTR | Rotate right |
| 1 | 0 | 1 | RTL | Rotate left |
| 1 | 1 | 0 | RRC | Rotate right through carry |
| 1 | 1 | 1 | RLC | Rotate left through carry |

    - If Constant_Control is 1:
      Ri = Rj (shifted or rotated [Constant] number of times)
    - Else:
      Ri = Rj (shifted or rotated Rk number of times)

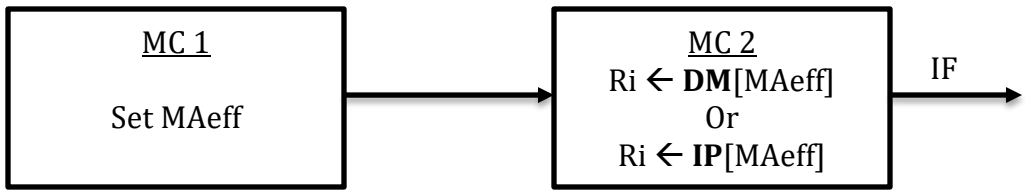
- **Floating Point (FPU) Instructions:**

  Uses same IW as the ALU instructions, however the operands are sent to the FPU instead of the ALU.
  - FA → Ri = Rj + Rk ; Floating point addition
  - FS → Ri = Rj - Rk ; Floating point subtraction
  - FM → Ri = Rj * Rk ; Floating point multiplication
  - FD → Ri = Rj / Rk ; Floating point division
  - FTI → Ri = int(Rj) ; Float to integer conversion
  - ITF → Ri = float(Rj) ; Integer to float conversion

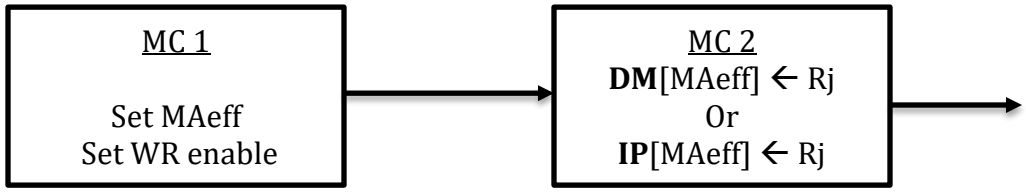
# Machine Cycles



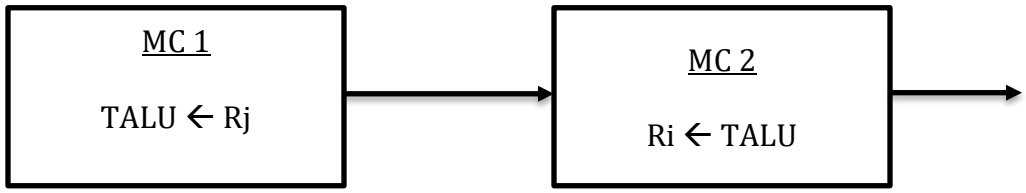

**Load:**



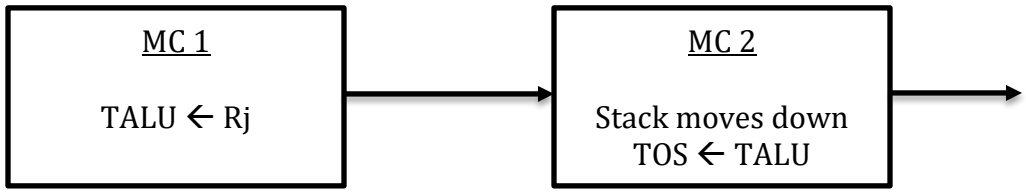

**Store:**



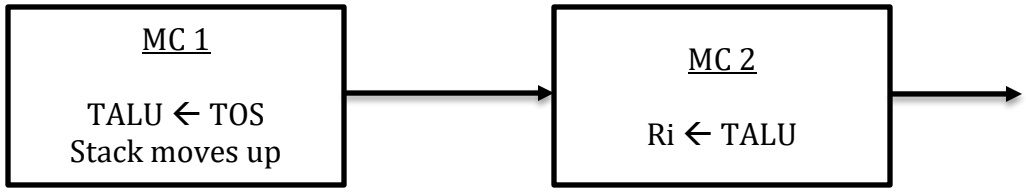

**Copy:**



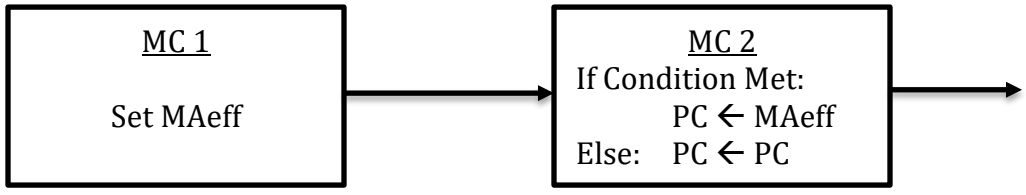

**Push:**



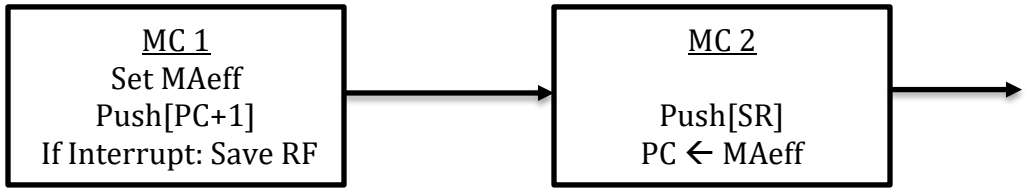

**Pop:**



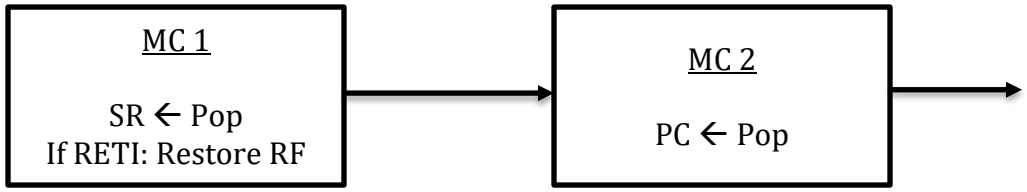

**Jump:**



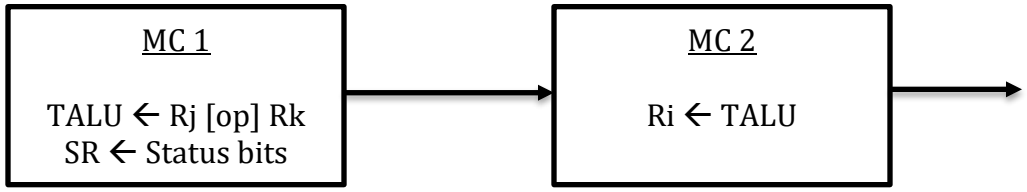
**Call:**



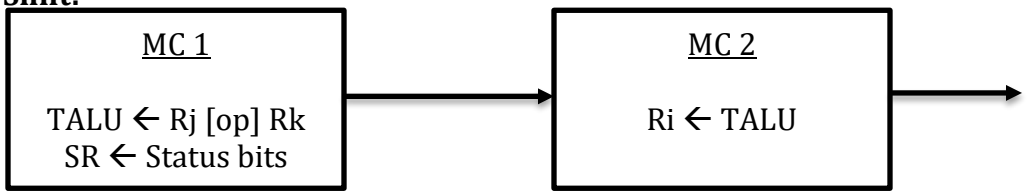

**Return:**




**ALU:**




**Rotate/Shift:**




**FPU:**