**Instructions:**

**EXT: [ Rd = (Rs OP2 Rt) ? 1 : 0 ] execution**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **EQ** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0000\_1000”** | **Rd** | **Rs** | **Rt** | **]** |
| **LT** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0000\_1001”** | **Rd** | **Rs** | **Rt** | **]** |
| **LE** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0000\_1010”** | **Rd** | **Rs** | **Rt** | **]** |
| **NE** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0000\_1011”** | **Rd** | **Rs** | **Rt** | **]** |

**EXT: [ Rd = Rs OP2 rt ]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ADD** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_0000”** | **Rd** | **Rs** | **Rt** | **]** |
| **AND** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_0100”** | **Rd** | **Rs** | **Rt** | **]** |
| **OR** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_0101”** | **Rd** | **Rs** | **Rt** | **]** |
| **XOR** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_0110”** | **Rd** | **Rs** | **Rt** | **]** |
| **SUB** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_1000”** | **Rd** | **Rs** | **Rt** | **]** |
| **NAND** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_1100”** | **Rd** | **Rs** | **Rt** | **]** |
| **NOR** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_1101”** | **Rd** | **Rs** | **Rt** | **]** |
| **NXOR** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0010\_1110”** | **Rd** | **Rs** | **Rt** | **]** |
| **RSHF** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0011\_0000”** | **Rd** | **Rs** | **Rt** | **]** |
| **LSHF** | **:** | **{opcode, Rd, Rs, Rt}** | **[** | **OP= “000000 0011\_0001”** | **Rd** | **Rs** | **Rt** | **]** |

**BR: [ if (Rs OP1 Rt) PC = PC + 4 + 4sxt(Imm) ]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **BEQ** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “001000 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **BLT** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “001001 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **BLE** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “001010 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **BNE** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “001011 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |

**JAL: [ Rt = PC+4, PC = Rs + 4sxt(Imm) ]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **JAL** | **:** | **{opcode, Rt, Imm(Rs)}** | **[** | **OP= “001100 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |

**LW: [ Rt = MEM[Rs+sxt(Imm) ]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **LW** | **:** | **{opcode, Rt, Imm(Rs)}** | **[** | **OP= “010010 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |

**SW: [ MEM[Rs+sxt(Imm) = Rt ]**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **SW** | **:** | **{opcode, Rt, Imm(Rs)}** | **[** | **OP= “011010 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |

**ALUI: Rt = Rs OP1 sxt(Imm)**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **ADDI** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “100000 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **ANDI** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “100100 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **ORI** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “100101 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |
| **XORI** | **:** | **{opcode, Rs, Rt, Imm}** | **[** | **OP= “100110 XXXX\_XXXX”** | **Imm** | **Rs** | **Rt** | **]** |

**Registers (Assembly | Register#):**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Zero** | **R0** |  | **A0** | **R1** |  | **A1** | **R2** |  | **A2** | **R3** |
| **A3/RV** | **R4** |  | **T0** | **R5** |  | **T1** | **R6** |  | **S0** | **R7** |
| **S1** | **R8** |  | **S2** | **R9** |  | **--** | **R10** |  | **--** | **R11** |
| **--** | **R12** |  | **FP** | **R13** |  | **SP** | **R14** |  | **RA** | **R15** |

**Pseudo-Instructions (Assembly | Equivalent Instruction):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **NOT Ri,Rj** | **NAND Ri,Rj,Rj** |  | **CALL Imm(Ri)** | **JAL RA,Imm(Ri)** |
| **RET** | **JAL R10,0(RA)** |  | **JMP Imm(Ri)** | **JAL R10,Imm(Ri)** |
| **BGT Ry,Rx,Label** | **BLT Rx,Ry,Label** |  | **BGE Ry,Rx,Label** | **BLE Rx,Ry,Label** |
| **BR Label** | **BEQ Zero,Zero,Label** |  | **GT Rz,Ry,Rx** | **LT Rz,Rx,Ry** |
| **GE Rz,Ry,Rx** | **LE Rz,Rx,Ry** |  | **SUBI Ry,Rx,Imm** | **ADDI Ry,Rx,–Imm** |

**Other notes:**

Memory addressability: Byte address

Instruction size: 4 Bytes (i.e., PC will be incremented by 4)

The data size for Load/Store: 4 Bytes

Data width for register file: 4 Bytes

Shift operations are arithmetic: (i.e. only RSHF needs sign extension)

**I/O: (not used in Pynq board)**

Clock (CLK), Reset signals

Memory mapped IO

ADDRLEDR: Memory Addr for LEDR: 0xFFFFF020

ADDRHEX: Memory Addr for HEX : 0xFFFFF000

ADDRKEY: Memory Addr for KEY : 0xFFFFF080

ADDRSW: Memory Addr for switch: 0xFFFFF090

**Microarchitecture specifications:**

Instruction memory: imem

Data memory: dmem

Start PC: 0x100