**1-Instructions format:**

16-bits instructions

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Family 2bit | Opcode 3bit | Rsrc 3bit | Rdst  3bit |
| MOV Rsrc, Rdst | 00 | 000 | Rsrc | Rdst |
| ADD Rsrc, Rdst | 00 | 001 | Rsrc | Rdst |
| SUB Rsrc, Rdst | 00 | 010 | Rsrc | Rdst |
| OR Rsrc, Rdst | 00 | 011 | Rsrc | Rdst |
| AND Rsrc, Rdst | 00 | 100 | Rsrc | Rdst |
| INC Rdst | 00 | 101 | Rdst | Rdst |
| DEC Rdst | 00 | 110 | Rdst | Rdst |
| NOT Rdst | 00 | 111 | Rdst | Rdst |
| OUT Rdst | 01 | 000 | Rdst | Rdst |
| IN Rdst | 01 | 001 | Rdst | Rdst |
| NOP | 01 | 010 | 000 | 000 |
| SETC | 01 | 101 | 000 | 000 |
| CLRC | 01 | 110 | 000 | 000 |
| PUSH Rdst | 10 | 000 | Rdst | Rdst |
| POP Rdst | 10 | 001 | Rdst | Rdst |
| LDD Rsrc, Rdst | 10 | 011 | Rsrc | Rdst |
| STD Rsrc, Rdst | 10 | 100 | Rdst | Rsrc |
| JZ Rdst | 11 | 000 | Rdst | Rdst |
| JN Rdst | 11 | 001 | Rdst | Rdst |
| JC Rdst | 11 | 010 | Rdst | Rdst |
| JMP Rdst | 11 | 011 | Rdst | Rdst |
| CALL Rdst | 11 | 100 | Rdst | Rdst |
| RET | 11 | 101 | 000 | 000 |
| RTI | 11 | 110 | 000 | 000 |

32-bits instructions

First 16-bits instructions are the op-code, Second 16-bits instructions are the immediate value

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Family 2bit | Opcode 3bit | Rsrc 3bit | Imm 16 bit |
| LDM Rdst, Imm | 10 | 010 | Rdst | #Imm |
| SHL Rsrc, Imm | 10 | 101 | Rsrc | #Imm |
| SHR Rsrc, Imm | 10 | 110 | Rsrc | #Imm |

**2-PC control unit:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| New PC | INTR signal | Pop\_signal  (pc\_to\_stack&memory\_read) | JMP\_signal  (branch&jump\_or\_not) | Stall\_signal |
| 32’b0(ISR) | 1 | X | X | X |
| Poped PC | 0 | 1 | X | X |
| Jump address | 0 | 0 | 1 | X |
| Current PC | 0 | 0 | 0 | 1 |
| Next pc | 0 | 0 | 0 | 0 |

**3-Entered Instruction in IF/IE buffer:**

1-If PC is freezed or the instruction needs Immediate value 🡺PC is freezed , Nop is Entered.

2-else it will be the output of instructions memory.

**4-Alu\_op (from decoding stage to ALU)**

|  |  |
| --- | --- |
| **alu\_op** | |
| NOP (MOV) | 0000 (MOV just pass the rs1) |
| ADD | 0001 |
| SUB | 0010 |
| AND | 0011 |
| OR | 0100 |
| INC | 0101 |
| DEC | 0110 |
| NOT | 0111 |
| SETC | 1000 |
| CLRC | 1001 |
| SHL | 1010 |
| SHR | 1011 |

**5-Jump type for the JMP detection unit:**

|  |  |  |
| --- | --- | --- |
| Jump type | MSB | LSB |
| jn | 0 | 0 |
| jz | 0 | 1 |
| jc | 1 | 0 |
| jmp | 1 | 1 |

**6-Control signals:**

|  |  |  |
| --- | --- | --- |
| Signal | # of bits | Value |
| alu\_op | 4 | Truth table 4 |
| imm | 1 | 1 in LDM,SHL &SHR instructions |
| pc\_to\_stack | 1 | 1 in CALL,RET&RTI instructions |
| alu\_src | 1 | 0 🡺Immediate, 1 🡺Rdst |
| mem\_write | 1 | 1 in STD,PUSH,CALL |
| mem\_read | 1 | 1 in LDM,LDD,RET,RTI&POP instructions |
| register\_write | 1 | 0 in STD,OUT,PUSH,CALL,RET,RTI,JZ,JN,JC&JMP |
| rti | 1 | 1 in RT instruction (to restore flags) |
| branch | 1 | 1 in JZ,JN,JC&JMP |
| Jump type | 2 | Truth table 5 |
| In\_port | 1 | 1 in IN (to write to the register from In port) |
| Out\_port | 1 | 1 in OUT (to enable out port) |
| mem\_to\_register | 1 | 1 in CALL |
| stack\_or\_data | 1 | 1 in PUSH,POP,CALL,RET&RTI |
| LDM | 1 | 1 IN LDM |

**7-Write back data source:**

|  |  |  |  |
| --- | --- | --- | --- |
| alu\_op | mem\_to\_register | in\_port | LDM |
| Memeory out | 1 | x | x |
| Alu\_out | 0 | 0 | 0 |
| In\_port\_data | 0 | 1 | x |
| immediate | 0 | 0 | 1 |

**8-Full forwarding unit truth table:**

|  |  |  |
| --- | --- | --- |
| ForwardA(out of FU)  First ALU operand | MSB | LSB |
| alu\_out\_IE\_MEM | 0 | 0 |
| alu\_out\_MEM\_WB | 0 | 1 |
| Rsrc (register file out) | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| ForwardB(out of FU)  Second ALU operand | MSB | LSB |
| alu\_out\_IE\_MEM | 0 | 0 |
| alu\_out\_MEM\_WB | 0 | 1 |
| Rdst (register file out) | 1 | 0 |

**9-EXCEPTIONS TYPES:**

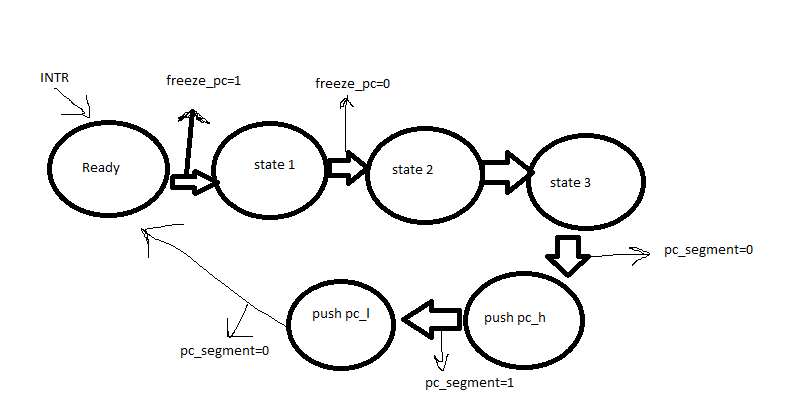
1-exception from dividing over zero or overflow (detected in execute stage).

10-When getting an interrupt:

stall (freeze pc)

push current PC to the stack

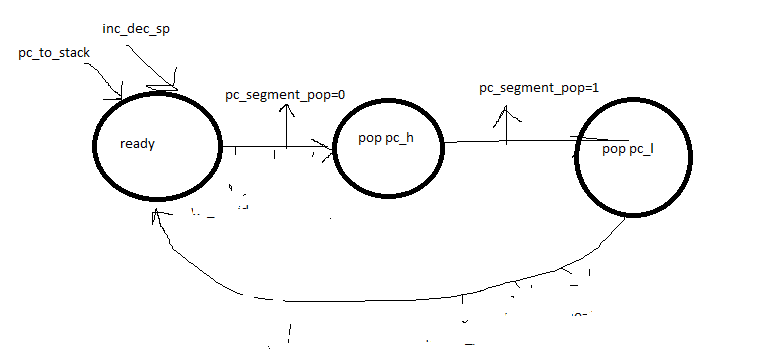
load the interrupt service routine address in the pc.



**11-When getting RTI or RET: after data memory (in memory stage):**

Pop PC from the stack

* PC already freezed in decoding stage.



|  |  |  |
| --- | --- | --- |
| Memory\_data | pc\_to\_stack | pc\_segment\_pop |
| alu\_out | 0 | 0 |
| alu\_out | 0 | 1 |
| pc\_h(higher part of PC) | 1 | 0 |
| pc\_l (lower part of PC) | 1 | 1 |