Logo

Description automatically generated

**Cairo University**

**Faculty of Engineering**

**Credit Hours System**

**Computer Architecture (CMPN301)**

**Report**

**Names:**

**Ali Said el Naggar  1180036**

**Abdallah Wael Marzouk   1180376**

**Mohamed Amr Afifi    1180062**

**Youssef Qadry Hashem 1180025**

**Instructions bit Details:**

**Identifier bit (indicates if this Instruction is 16 or 32 bits (0 for 16, 1 for 32))**

**One Operand:**

Bit 31-30: Operation Type

Bit 29-27: Op Code for Function

Bit 26-24: Register Address

Bit 23-17: Not Used

Bit 16: Identifier bit

Bit 15-0: Not Used

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **One Operand** | | | | | |
| **Instruction Bit** | **Bits 31-30** | **Bits**  **29-27** | **Bits 26-24** | **Bits 23-17** | **Bit 16** | **Bits 15-0** |
| NOP | 00 | 000 | Not Used | Not Used | 0 | Not Used |
| SETC | 00 | 001 | Not Used | Not Used | 0 | Not Used |
| CLRC | 00 | 010 | Not Used | Not Used | 0 | Not Used |
| NOT Rdst | 00 | 011 | Rdst | Not Used | 0 | Not Used |
| INC Rdst | 00 | 100 | Rdst | Not Used | 0 | Not Used |
| DEC Rdst | 00 | 101 | Rdst | Not Used | 0 | Not Used |
| OUT Rdst | 00 | 110 | Rdst | Not Used | 0 | Not Used |
| IN Rdst | 00 | 111 | Rdst | Not Used | 0 | Not Used |

**Two Operands:**

Bit 31-30: Operation Type

Bit 29-27: Op Code for Function

Bit 26-24: Rdst Register Address

Bit 23-21: Rsrc Register Address / 23-19 Shift Amount

Bit 18-17: Not Used

Bit 16: Identifier bit

Bit 15-0: Immediate value for Add

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Two Operand** | | | | | | |
| **Instruction Bit** | **Bits**  **31-30** | **Bits**  **29-27** | **Bits 26-24** | **Bits**  **23-21/23-19** | **Bits**  **18-17** | **Bit**  **16** | **Bits**  **15-0** |
| Mov Rsrc, Rdst | 01 | 000 | Rdst | Rsc | Not  Used | 0 | Not  Used |
| ADD Rsrc, Rdst | 01 | 001 | Rdst | Rsc | Not  Used | 0 | Not  Used |
| IADD Rdst, Imm | 01 | 010 | Rdst | Not  Used | Not  Used | 1 | Immediate value |
| SUB Rsrc, Rdst | 01 | 011 | Rdst | Rsc | Not  Used | 0 | Not  Used |
| AND Rsrc, Rdst | 01 | 100 | Rdst | Rsc | Not  Used | 0 | Not  Used |
| OR Rsrc, Rdst | 01 | 101 | Rdst | Rsc | Not  Used | 0 | Not  Used |
| SHL Rsrc, Imm | 01 | 110 | Rdst | Shift amount | Not  Used | 1 | Immediate value |
| SHR Rsrc, Imm | 01 | 111 | Rdst | Shift amount | Not  Used | 1 | Immediate value |

**Bit 15-0: Immediate value for Add**

**Memory Operations:**

Bit 31-30: Operation Type

Bit 29-27: Op Code for Function

Bit 26-24: Rdst Register Address

Bit 23-21: Rsrc Register Address

Bit 20-17: Not Used

Bit 16: Identifier bit

Bit 15-0: Immediate value for LDM to Register / Offset Address

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Memory Operand** | | | | | | |
| **Instruction Bits** | **Bits 31-30** | **Bits 29-27** | **Bits 26-24** | **Bits 23-21** | **Bits**  **20-17** | **Bit 16** | **Bits**  **15-0** |
| PUSH Rdst | 10 | 000 | Rdst | Not Used | Not Used | 0 | Not Used |
| POP Rdst | 10 | 001 | Rdst | Not Used | Not Used | 0 | Not Used |
| LDM Rdst, Imm | 10 | 010 | Rdst | Not Used | Not Used | 1 | Used |
| LDD Rdst, offset (Rsrc) | 10 | 011 | Rdst | Rsrc | Not Used | 1 | Used |
| STD Rsrc1, offset (Rsrc2) | 10 | 100 | Rdst | Rsrc | Not Used | 1 | Used |

**Branch and Change of Control Operations:**

Bit 31-30: Operation Type

Bit 29-27: Op Code for Function

Bit 26-24: Rdst Register Address

Bit 23-17: Not Used

Bit 16: Identifier bit

Bit 15-0: Not Used

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Instruction** | **Branch and Change of Control Operations** | | | | | |
| **Instruction Bits** | **Bits**  **31-30** | **Bits**  **29-27** | **Bits**  **26-24** | **Bits**  **23-17** | **Bit**  **16** | **Bits**  **15-0** |
| JZ Rdst | 11 | 000 | Rdst | Not Used | 0 | Not Used |
| JN Rdst | 11 | 001 | Rdst | Not Used | 0 | Not Used |
| JC Rdst | 11 | 010 | Rdst | Not Used | 0 | Not Used |
| JMP Rdst | 11 | 011 | Rdst | Not Used | 0 | Not Used |
| Call Rdst | 11 | 100 | Rdst | Not Used | 0 | Not Used |
| RET | 11 | 101 | Not Used | Not Used | 0 | Not Used |

**Reset Signal:**

When Reset = 0 , first mux of pc will make PC = 0 and output M[0]

**Control unit (23 bits)**

* Receive first 5 bits from instruction to Decide which control signals should be sent  
  Signals are:  
  1-  Memory Read enable (1 bit)  
  2 - Memory Write enable (1 bit)  
  3- Register Write enable (1 bit)  
  4- IF/D flush (1 bit)  
  5- ID/EX flush (1 bit)  
  6- Multiplexer Selector "Write Back Value" to choose between Data from memory or ALU result (1 bits)  
  7- Stack signal (5 bits) which are :
  + SP enable (1 bit)
  + SP Add or OLD (POP or push)
  + SelectorSP (1 bit)
  + SP or ALU result (Memory stage)
  + PC + 1 or Read data 1 (Memory stage)

8- Signal for Forwarding unit for other operands ( 2 bits) one for each register  
  This signal controls multiplexers at ALU since Control unit has only 2 options: read data 1 or data 2 while forward unit has the other 2 options.  
 9- ALU operation Selector (4 bits)  
10- Input and output Ports selector (2 bits)

* 1 bit for output port enable
* 1 bit for input port or ALU result

11- Jump Signals (3 bits)

* (1 bit) signal for flag enable
* (2 bits) jump signals  for 4 jumps(zero,carry,negative,unconditional)

           12 - Shift or immediate (1 bit)  
           13- PC selector (2 bits)

**Pipeline Registers details:**

|  |  |  |
| --- | --- | --- |
| **Register** | **Size** | **Input** |
| IF/ID | 98 bits | Input Port (32 bit)  PC (32 bit)  Instruction (32 bit)  Enable (1 bit)  Flush Signal (1 bit) |
| ID/EX | 186 bits | Input Port (32 bit)  PC (32 bit)  Control Signal (18 bit)  Read Data 1 (32 bit)  Read Data 2 (32 bit)  Extended Imm/Offset/ Shift (32 bit)  Address for Rdst1 (3 bit)  Address for Rsrc (3 bit)  Enable (1 bit)  Flush Signal (1 bit) |
| EX/MEM | 110 bits | PC (32 bit)  Control Signal (10 bit)  Read Data 1 (32 bit)  ALU Result (32 bit)  Address for Rdst1 (3 bit)  Enable (1 bit) |
| MEM/WE | 102 bits | PC (32 bit)  Control Signal (2 bit)  Read Data 1 (32 bit)  ALU Result (32 bit)  Address for Rdst1 (3 bit)  Enable (1 bit) |

**Pipeline Hazards:**

**Data Hazards:**

Hazards due to register dependencies. When an instruction cannot be executed because it depends on an earlier one that doesn’t hold the correct value yet.

**Diagram

Description automatically generated with medium confidence**

**Control Hazards:**

The control hazard faced in the design was when the branch instructions check for the flags - in the Execution stage- of the previous instruction, which isn’t available. The Solution was to use Static Branch Prediction.

Static Branch Prediction (Predict not taken):

Automatically, the PC fetches the next instruction when the current one is in the decoding stage, it will remain until the execution stage of the current instruction.

After checking for the flags in the execution stage, if it’s true, IF/ID and ID/EX buffers will flush and the PC will be changed to the new instruction. If it’s false, the instruction will remain to the next stages

The unconditional jumps don’t require checking for flags. Therefore, it doesn’t wait for the execution and always sends the PC the address in the decode stage and the control unit adjusts the PC selector. So, unconditional jumps don’t require prediction.

In RET, we will have to flush and stall the pipe 2 cycles till we get the PC value from the memory (*Stack*).

**Hazards Solutions:**

1. **Forwarding Unit:**

The forwarding unit will check if the data (Rsrc) in this stage needs a data in (Rdst) from the previous stage(Read After Write)if the previous condition is true, the forwarding will take value from ALU or Memory to use it in the ALU, else the control unit will send a signal to the forwarding unit to choose another values in the ALU which are (Read data 1, Read data 2 or Immediate value)

1. **Hazard Detection Unit:**

Load Use Case,

If (ID/EX.MemRead ==1)

If (ID/EX.Rs2==IF/ID.Rs1) or (ID/EX.Rs2==IF/ID.Rs2)

{ PC Selector=0 Stall IF Stall ID }