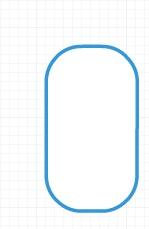
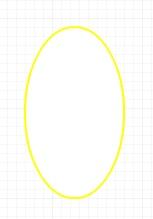
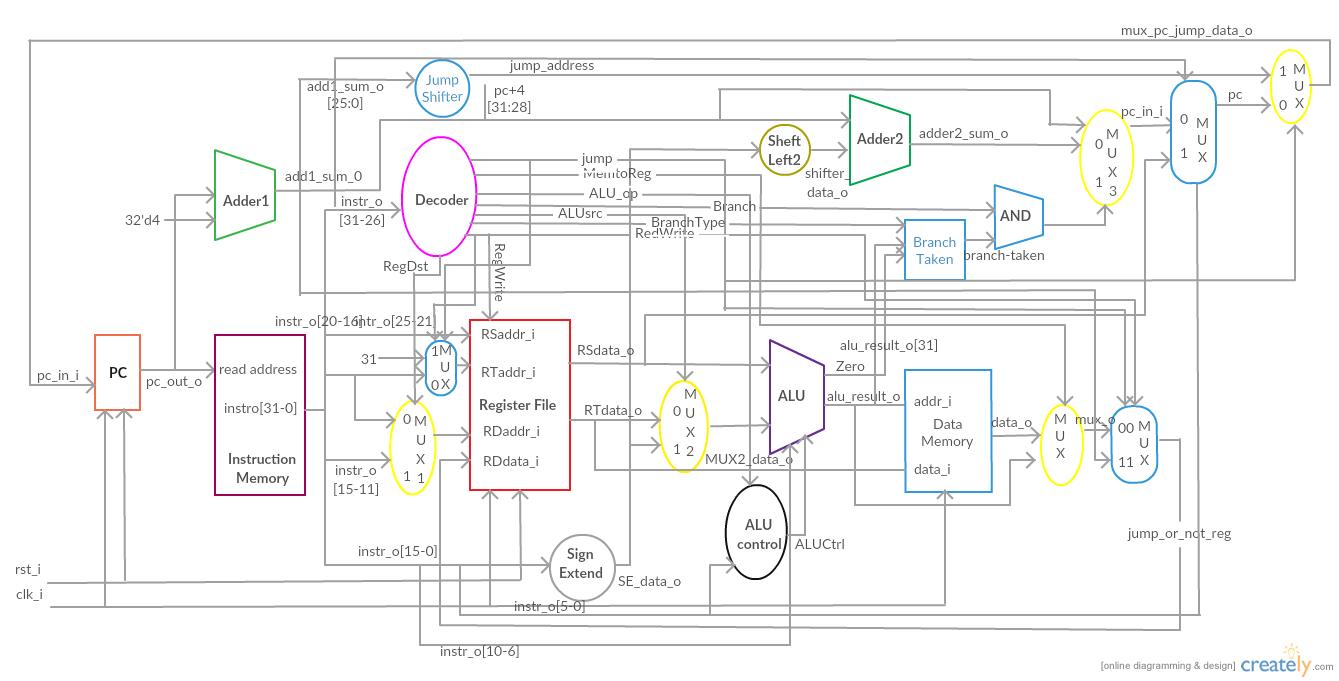
**Computer Organization**



**Architecture diagram:**

**Detailed description of the implementation:**

|  |  |
| --- | --- |
| **mul** | r-type, funct\_i = 24 時做乘法 |
| **lw** | i-type rs做加法得出的值為address  address input至data memory 找出要load/sw 之值/位置  再寫回rt中 ; 或是將rt存的值寫入data memory |
| **jump** | jump shifter 會將instr[25:0]往左移2位  再assign pc+4[31:28]配上此address 為 jump address |
| **jal** | 會存要return 回去的address到第31個reg  存的address為pc+4  [assign jal\_or\_not\_reg]指令便是用來判斷是不是jal來決定address要是pc+4或是alu算出之address |
| **jr** | jump 到reg(rs)所存之位址  直接用[assign pc]來判斷是不是jr決定下一條指令是rs所存之值或是經過運算出之位址(oc\_in\_i) |
| **ble** | branch type 為1 ,由branch taken component 來判斷branch taken or not,結果再與 branch and 在一起來決定是否branch |
| **bnez** | branch type 為0 ,由branch taken component 來判斷branch taken or not,結果再與 branch and 在一起來決定是否branch |
| **bltz** | branch type 為2 ,由branch taken component 來判斷branch taken or not,結果再與 branch and 在一起來決定是否branch |

**Problems encountered and solutions:**

**Lesson learnt (if any):**