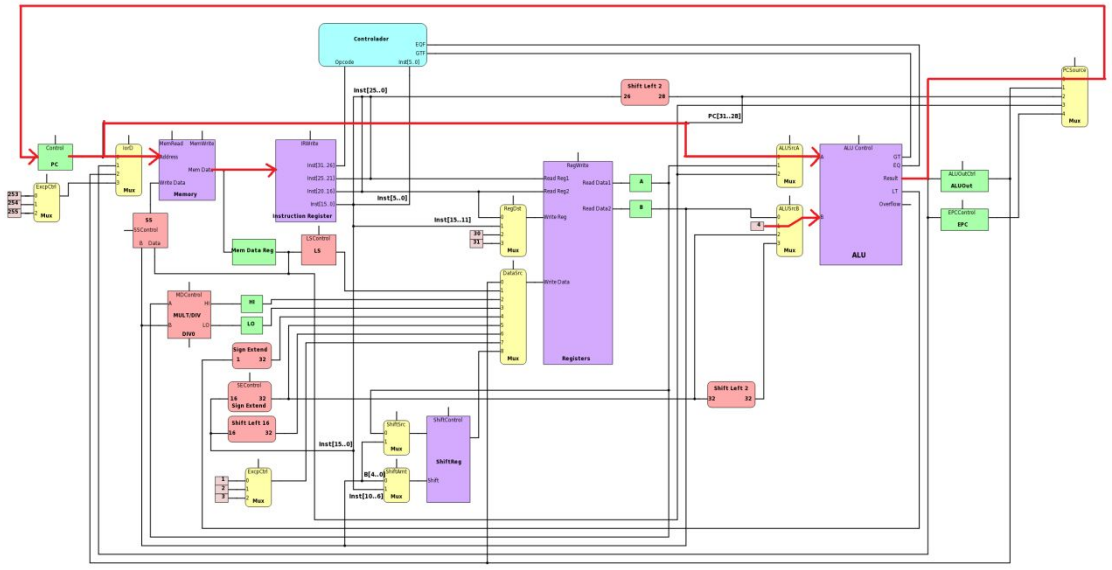
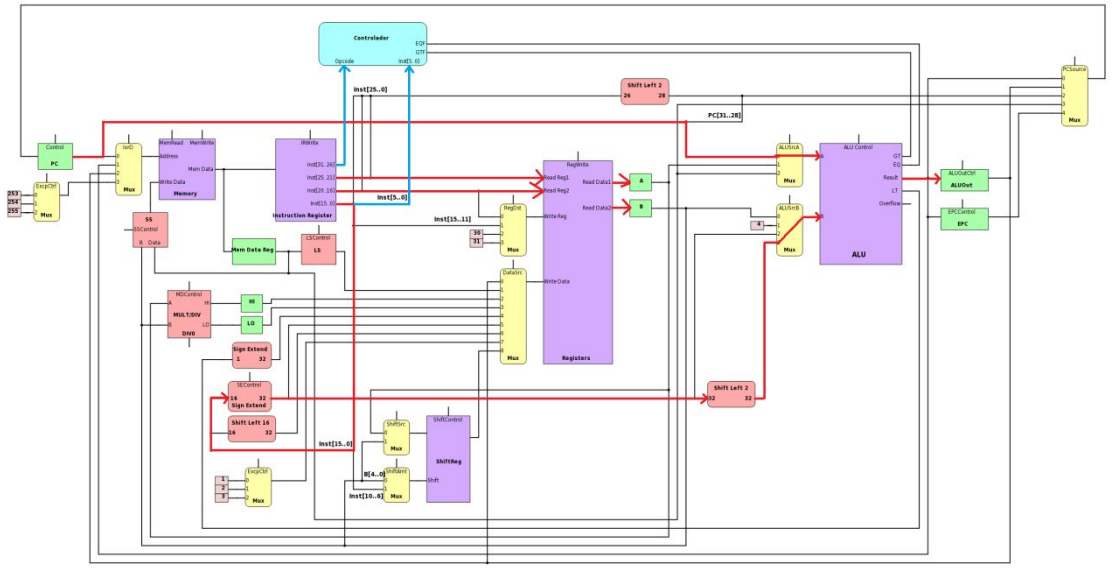
Todas as operações:

1. Lê memória e incrementa PC

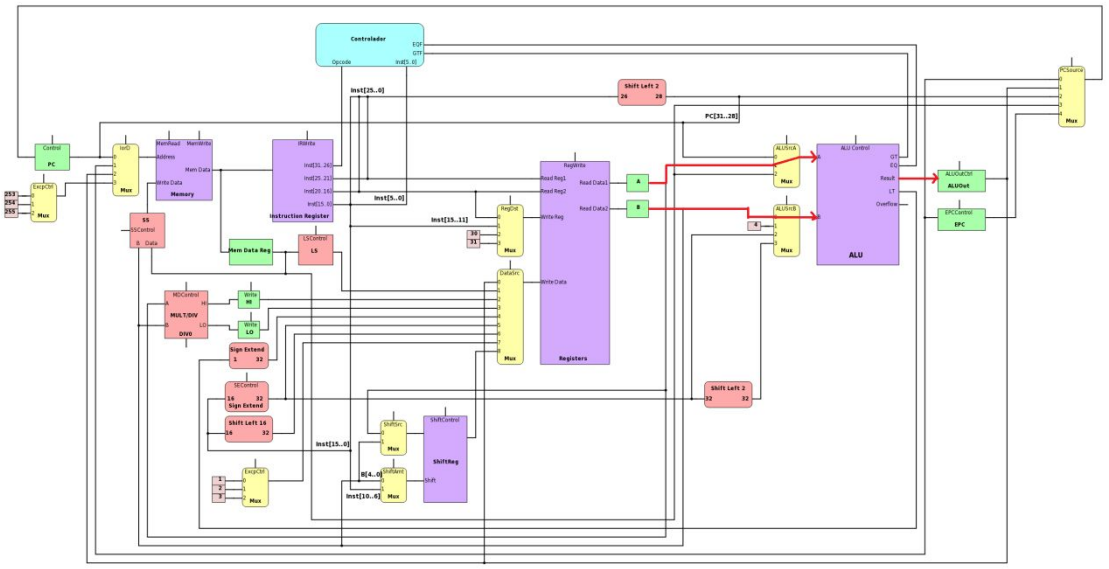


2.Carrega valor dos registradores em A e B e decodifica branch

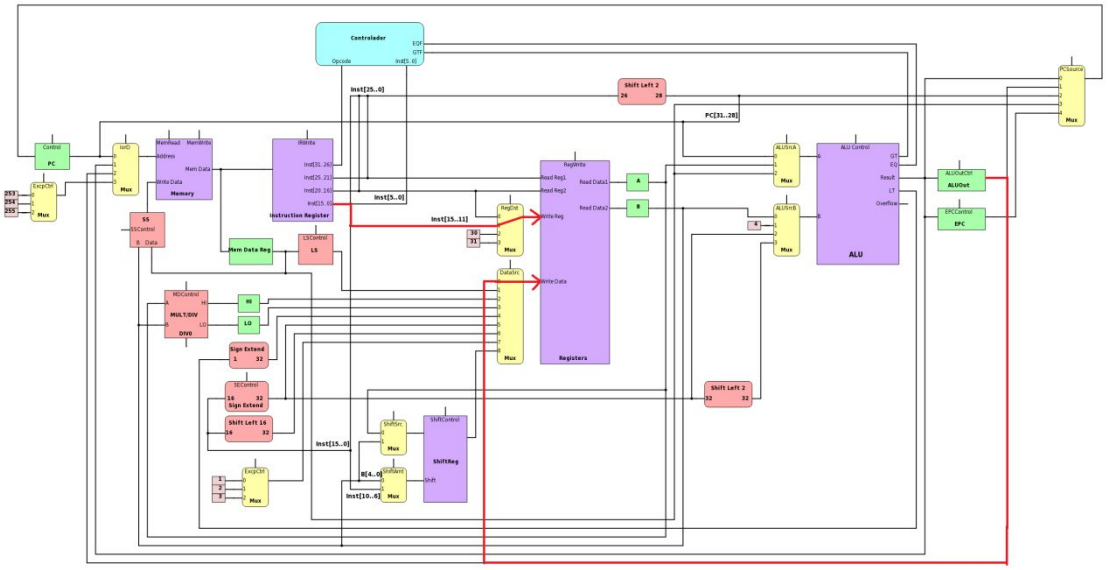


ADD rd, rs, rt | rd <- rs + rt
AND rd, rs, rt | rd <- rs & rt
SUB rd, rs, rt | rs <- rs - rt

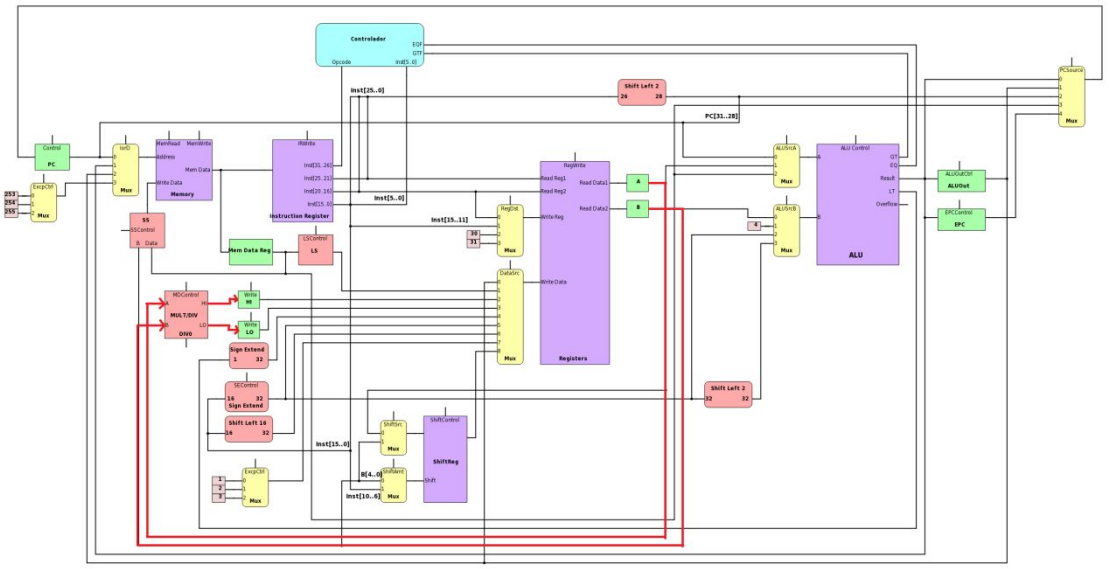
3.Realiza operação e grava em ALUOut



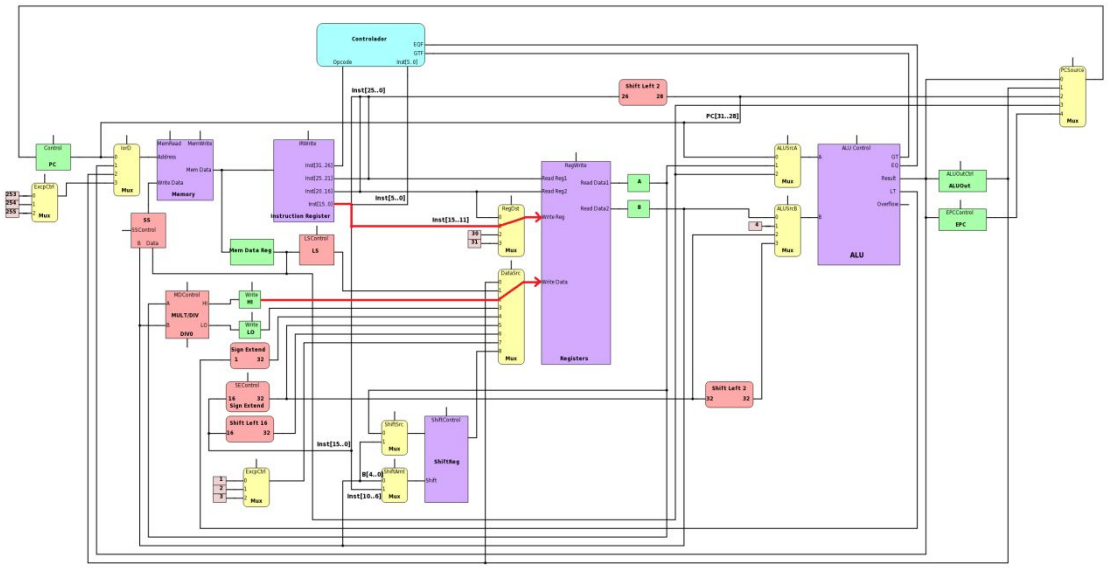
4.Grava ALUOut no registrador

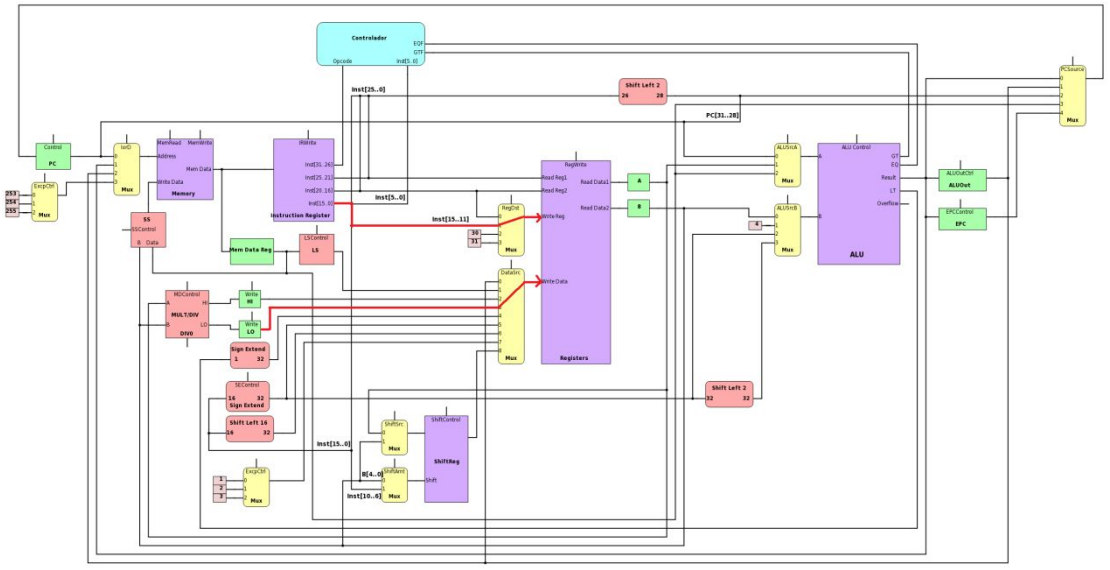


DIV rs, rt
MULT rs, rt
3.Realiza operação e grava resultado nos registradores HI e LO



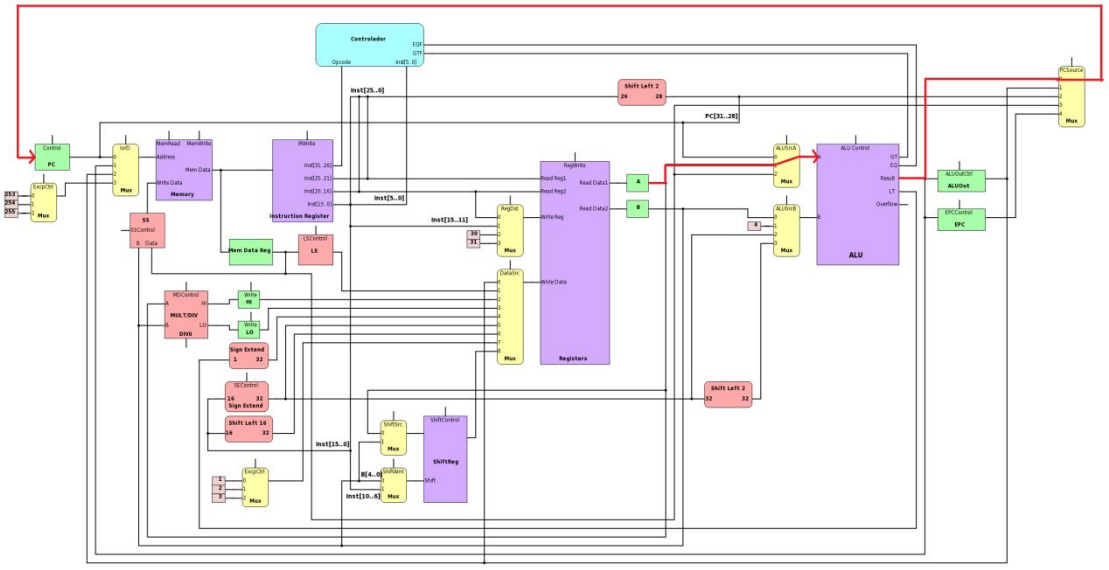
MFHI rd
MFLO rd
3.Grava o valor em HI ou LO em rd



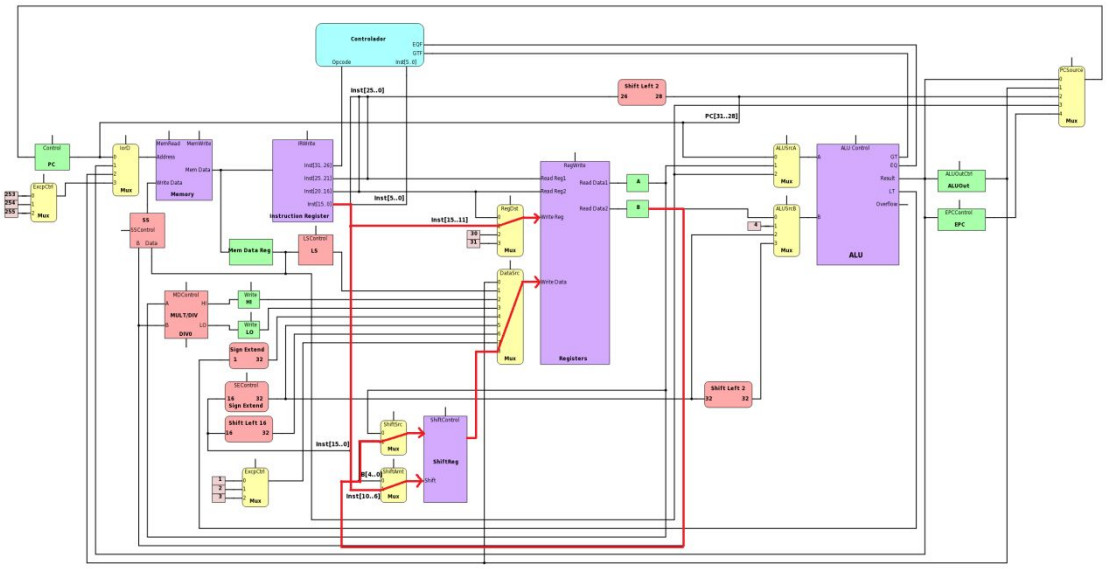


JR rs

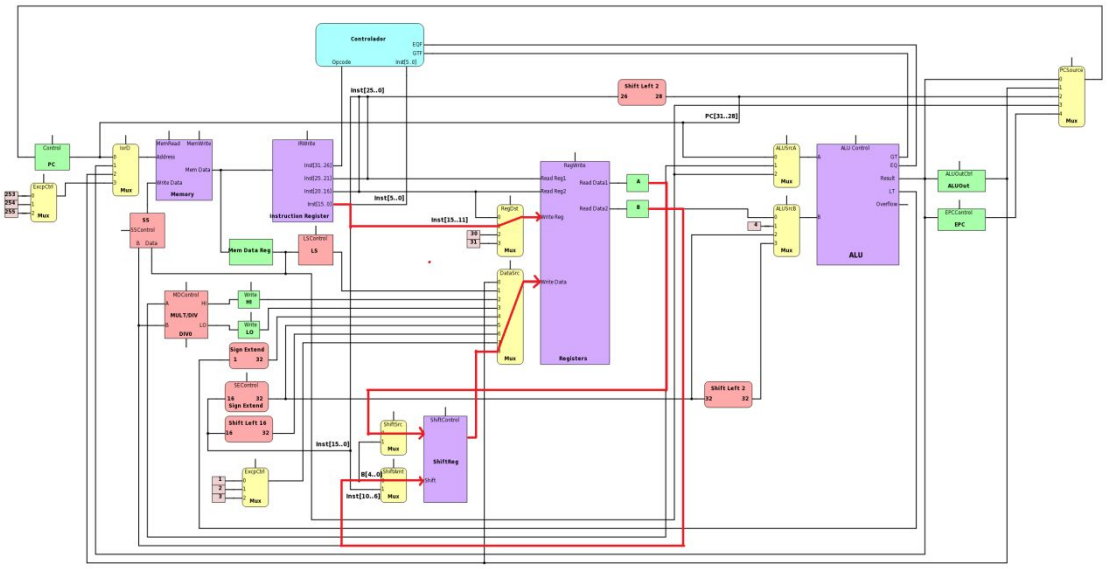
3.Grava o endereço que está em rs no PC



SLL rd, rt, shamt | rd <- rt << shamt
SRA rd, rt, shamt | rd <- rt >> shamt*
SRL rd, rt, shamt | rd <- rt >> shamt
3.Desloca o valor do registrador B em ‘shamt’ unidades e grava em rd

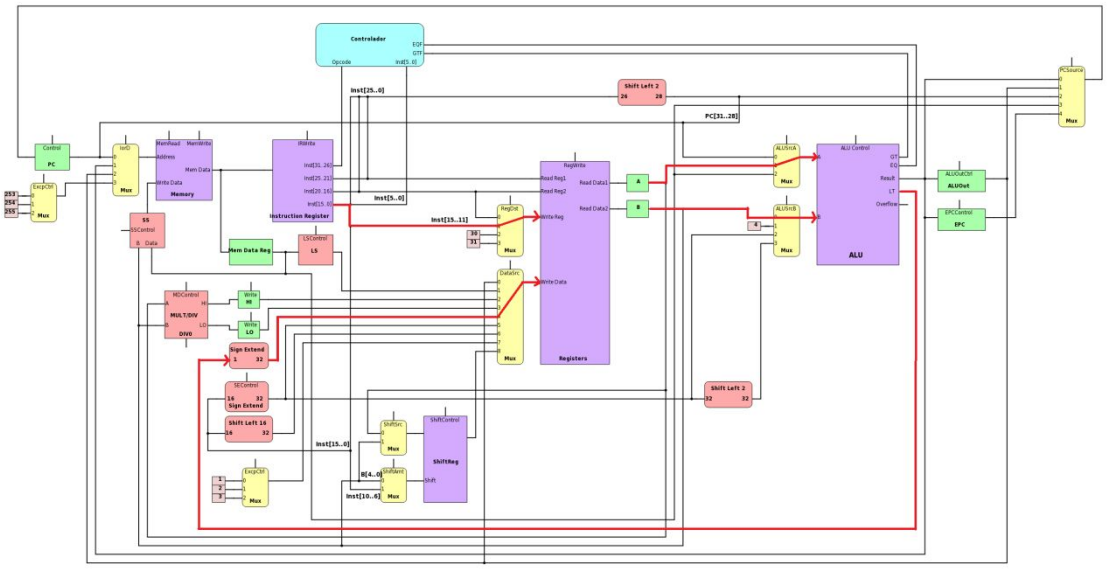


SLLV rd, rs, rt | rd <- rs << rt
SRAV rd, rs, rt | rd <- rs >> rt*
3. Desloca o valor do registrador A uma quantidade definida pelo registrador B e grava em rt



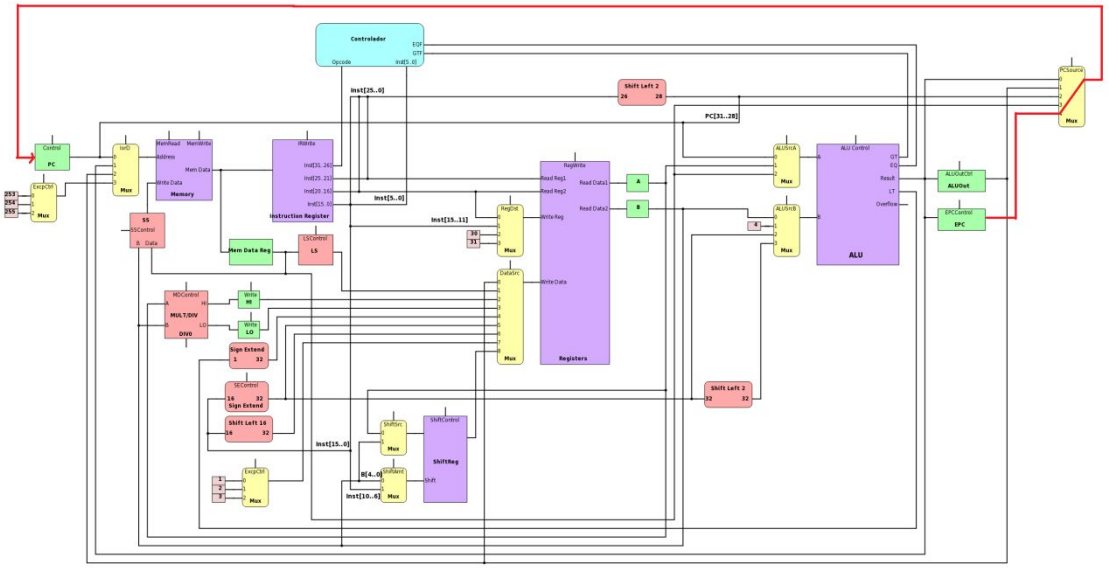
SLT rd, rs, rt | rd <- (rs < rt) ? 1 : 0

3.Compara o valor do registrador A com o valor de B e joga o valor da flag LT estendido para 32 bits em rd



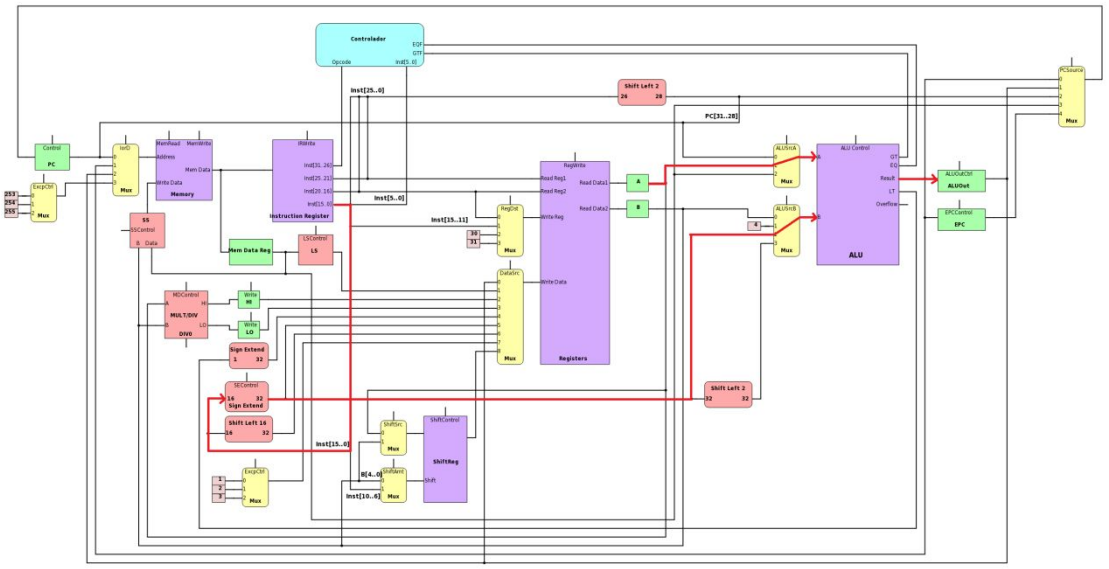
RTE | PC <- EPC

3. Carrega o valor do registrador EPC em PC

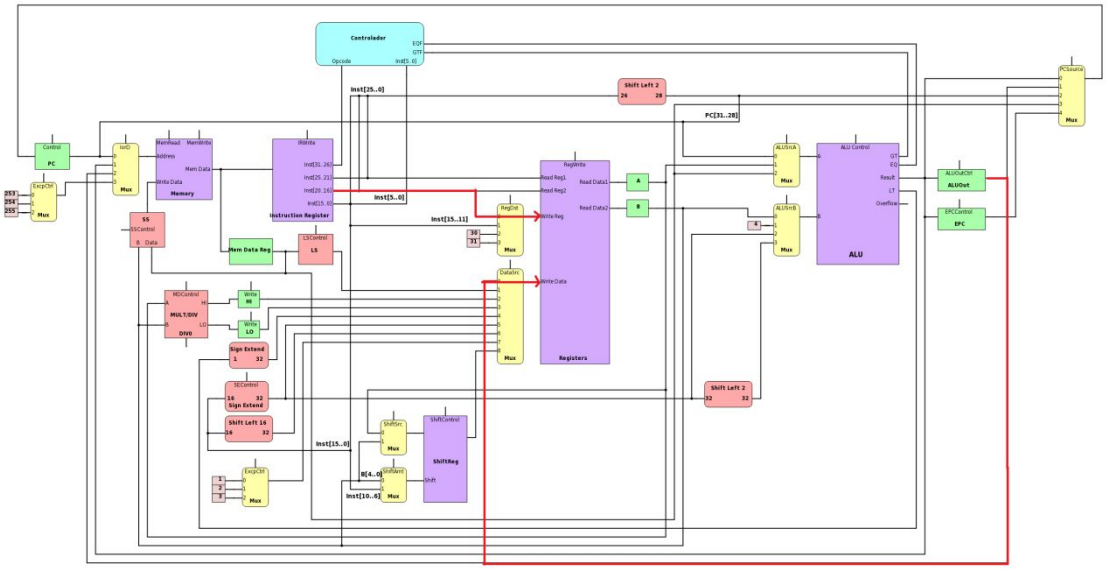


ADDI rt, rs, immediate | rt <- rs + immediate
ADDIU rt, rs, immediate | rt <- rs + immediate

3.Calcula a soma do valor no registrador A com o numero ‘immediate’ estendido para 32 bits, e salva em ALUOut

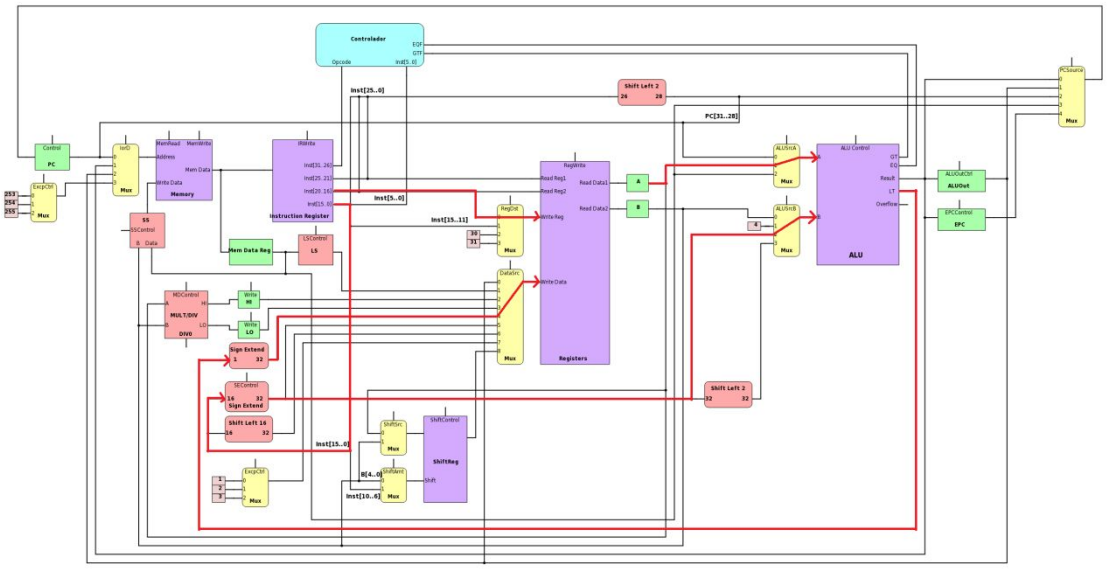


4.Grava o valor de ALUOut em rt



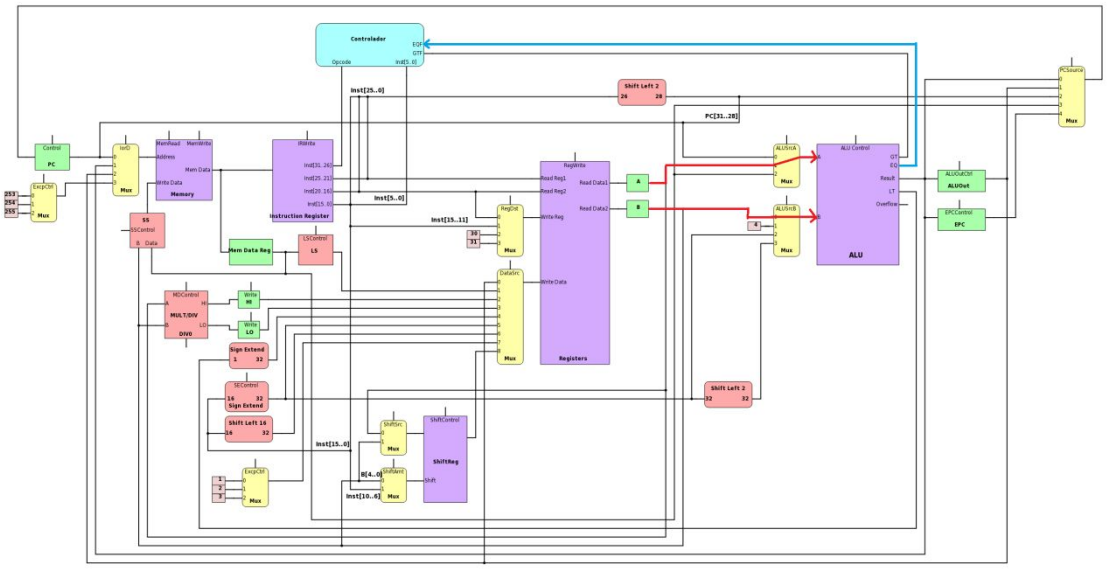
SLTI rt, rs, immediate | rt <- (rs < immediate) ? 1 : 0

3.Compara o valor do registrador A com o valor de ‘immediate’ estendido para 32 bits e joga o valor da flag LT estendido para 32 bits em rt

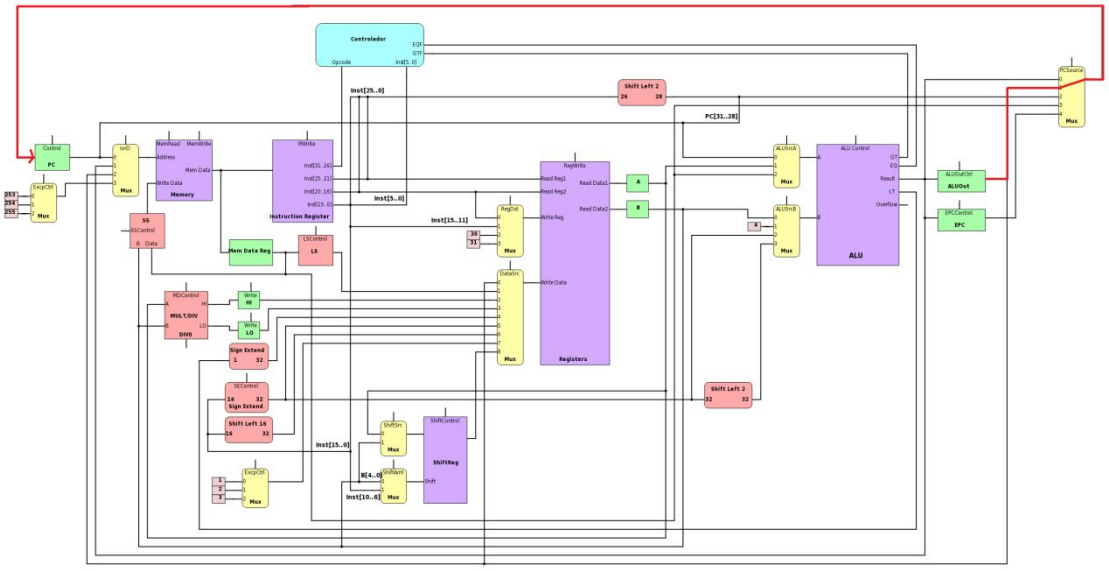


BEQ rs, rt, offset | Desvia se rs == rt
BNE rs, rt, offset | Desvia se rs != rt

3. Compara os valores dos registradores A e B e envia o resultado da flag EQ para o controlador

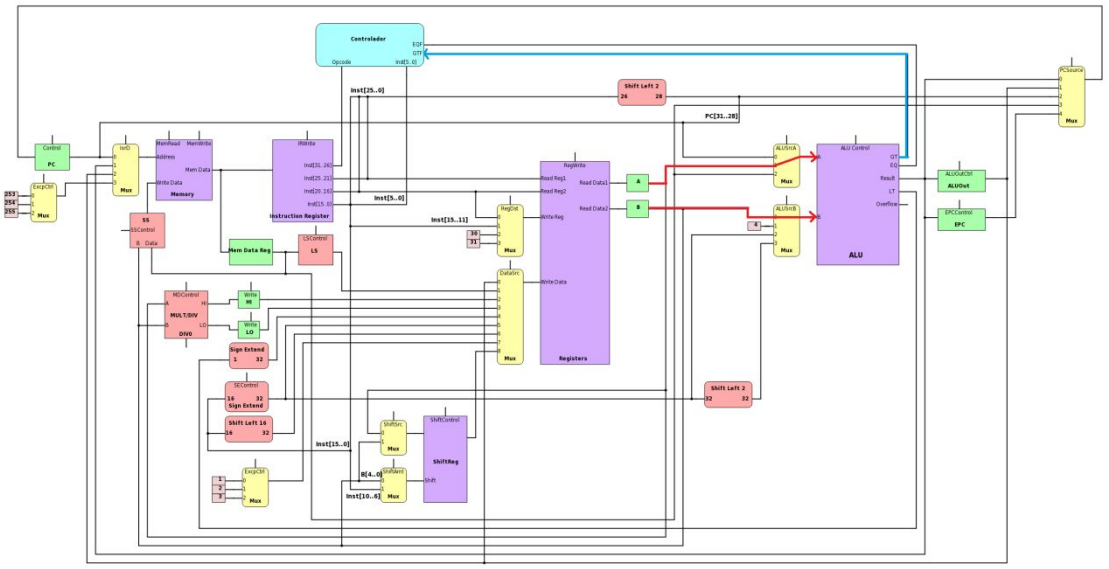


4. Grava o valor em ALUOut (calculado no segundo clock, ver primeira pagina) em PC caso EQ == 1 (no BEQ) ou EQ == 0 (no BNE)

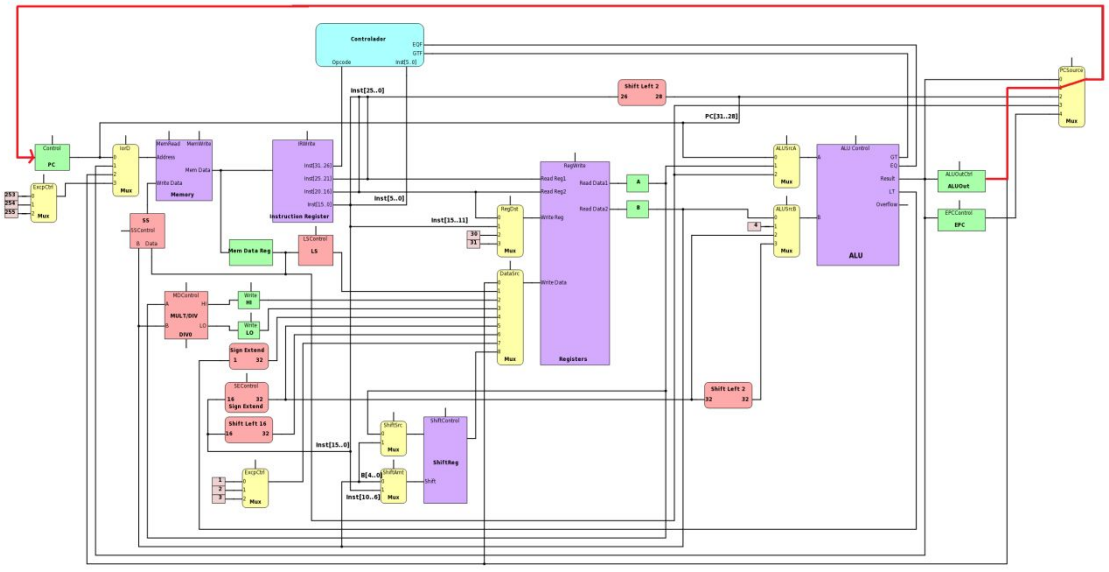


BLE rs, rt, offset | Desvia se rs <= rt
BGT rs, rt, offset | Desvia se rs > rt

3. Compara os valores dos registradores A e B e envia o resultado da flag GT para o controlador

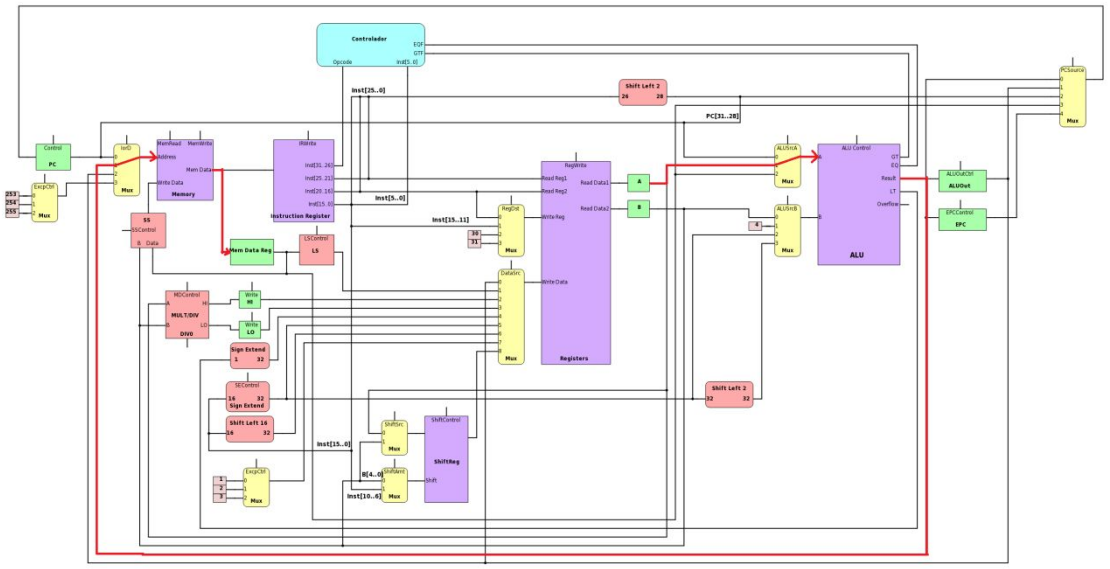


4. Grava o valor em ALUOut (calculado no segundo clock, ver primeira pagina) em PC caso GT == 1 (no BGT) ou GT == 0 (no BLE)

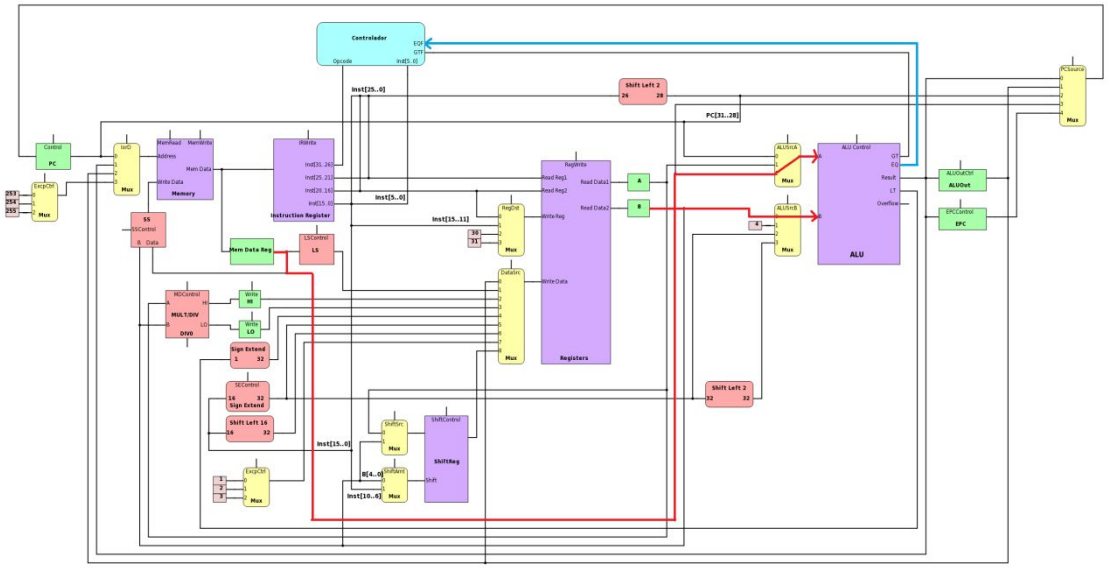


BEQM rs, rt, offset | Desvia se Mem[rs] == rt

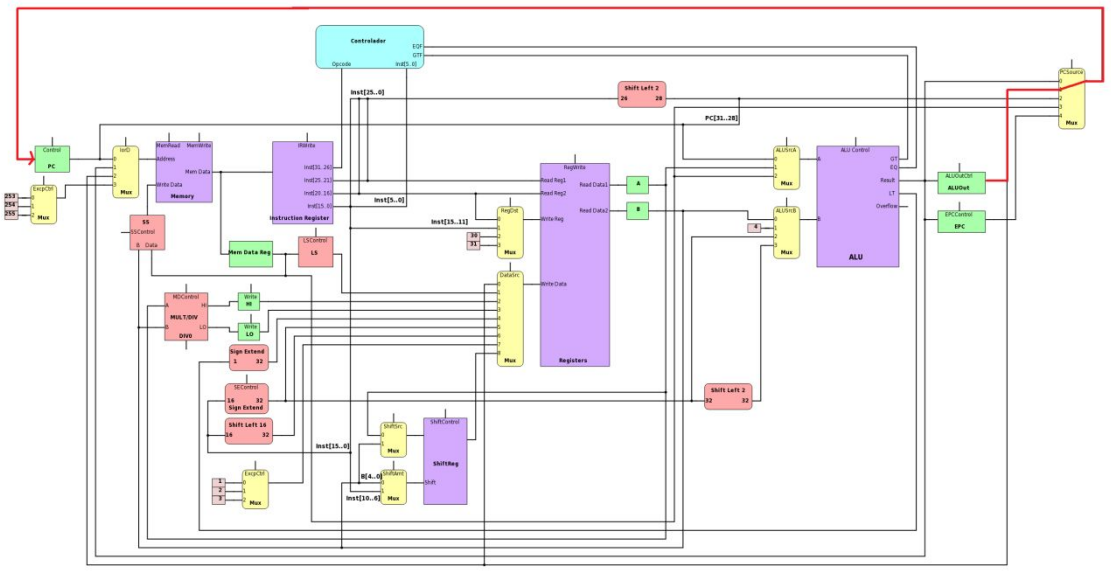
3.Busca o valor na memória, na posição especificada no registrador A, e grava em MDR



4.Compara o valor em MDR com o valor no registrador B e envia o resultado da flag EQ para o controlador

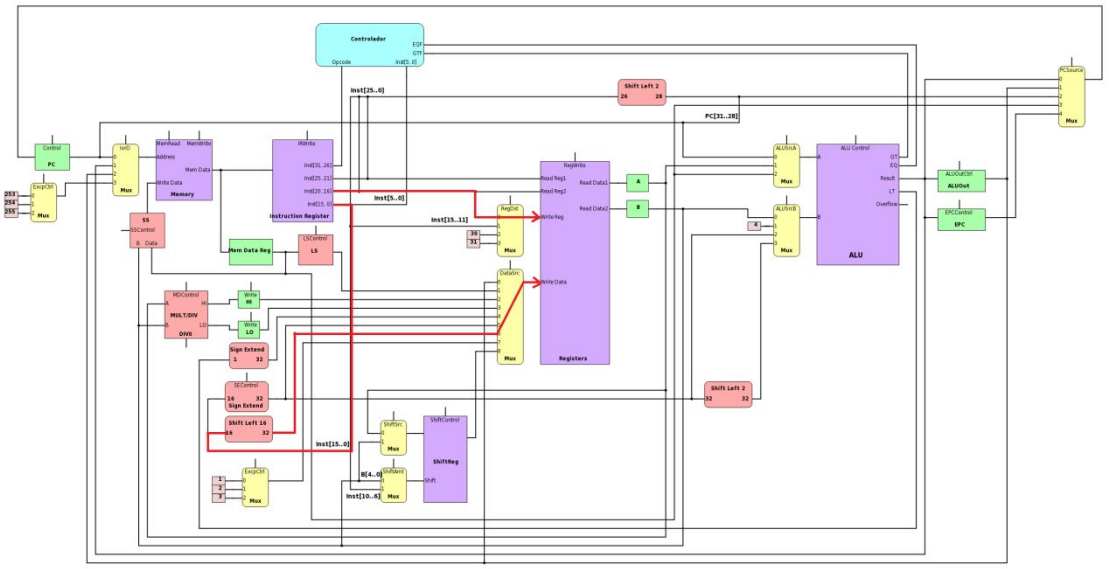


5. Grava o valor em ALUOut (calculado no segundo clock, ver primeira pagina) em PC caso EQ == 0

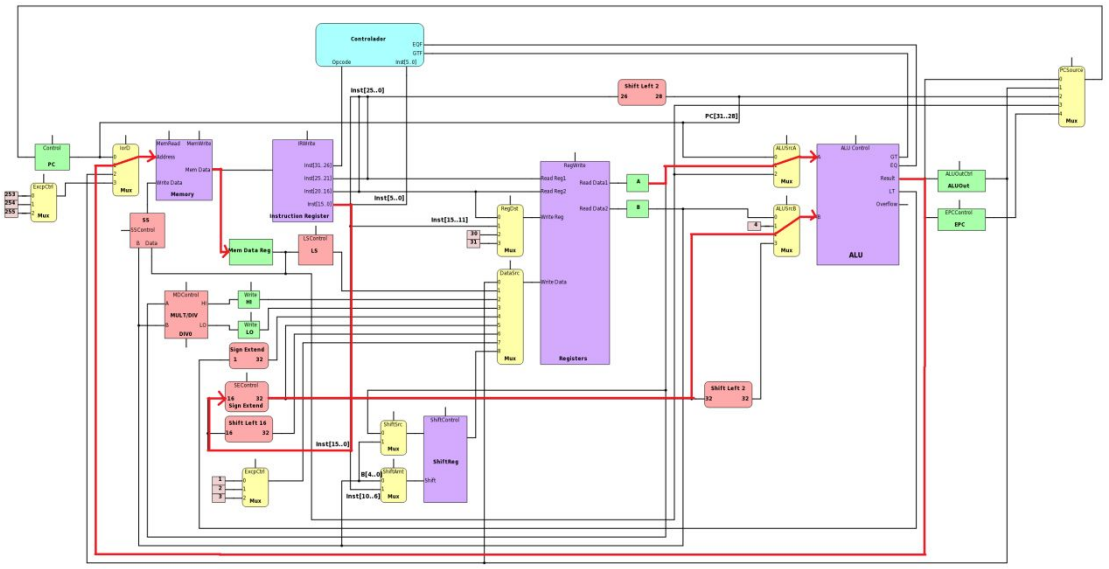

Controlador
EQT
GTT
Opcode
Inst[5..0]
Inst[25..0]
Shift Left 2
26
28
PCSource
Mux
PC[31..28]
Control
PC
ExcpCtrl
Mux
IorD
Mux
MemRead
MemWrite
Address
Mem Data
Write Data
Memory
IRWrite
Inst[31..26]
Inst[25..21]
Inst[20..16]
Inst[15..0]
Instruction Register
Inst[5..0]
Inst[15..11]
RegDst
Mux
30
31
RegWrite
Read Reg1
Read Reg2
Write Reg
Read Data1
Read Data2
Registers
A
B
ALUSrcA
Mux
ALUSrcB
Mux
4
ALU Control
GT
EQ
Result
LT
Overflow
ALU
ALUOutCtrl
ALUOut
EPCControl
EPC
SS
SSControl
R Data
Mem Data Reg
LSControl
LS
DataSrc
Mux
MDControl
HI
MULT/DIV
LO
DIVO
Write HI
Write LO
Sign Extend
1 32
SEControl
16 32
Sign Extend
Shift Left 16
16 32
Shift Left 2
32 32
ShiftSrc
Mux
ShiftControl
ShiftReg
Shift
Inst[15..0]
R[4..0]
Inst[10..6]
ShiftAmt
Mux
1
2
3
ExcpCtrl
Mux

LUI rt, immediate | rt <- immediate << 16

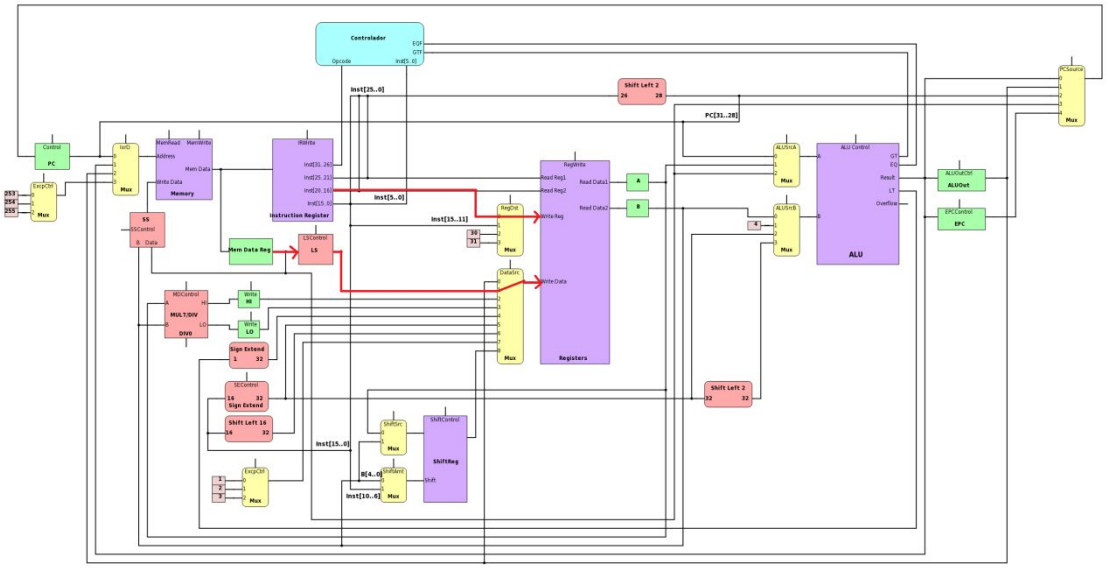
3.Grava o valor de ‘immediate’ deslocado em 16 bits para a esquerda em rt



LW rt, offset(rs) | rt <- Mem[rs + offset]
LH rt, offset(rs) | rt <- halfword Mem[rs + offset]
LB rt, offset(rs) | rt <- byte Mem[rs + offset]
3.Calcula a soma de rs com ‘offset’ estendido para 32 bits, e salva a informação deste endereço da memória em MDR

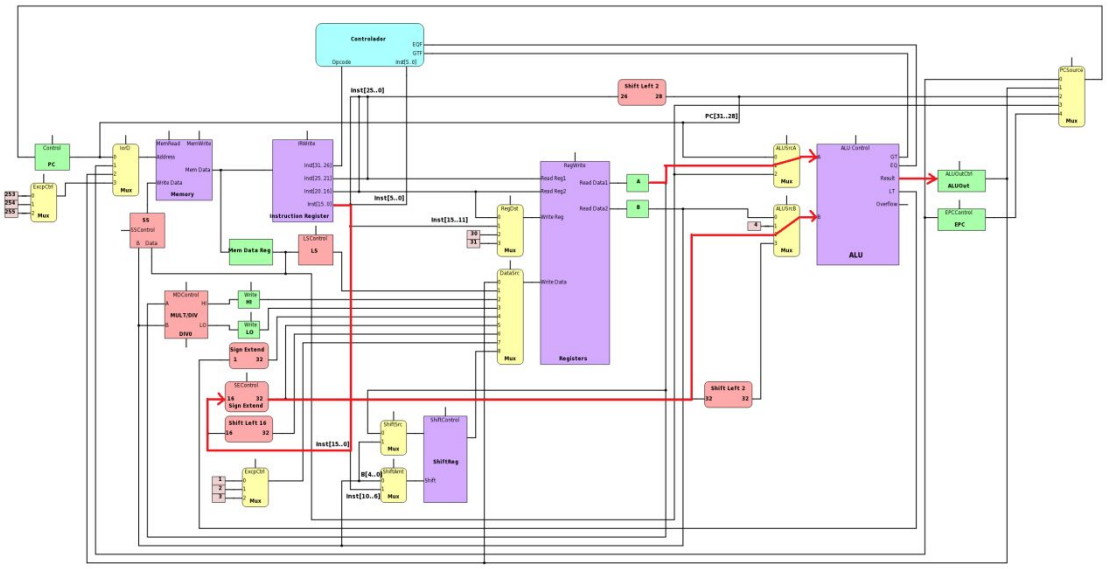


4.Grava o valor de MDR, após passar por LS(Load Size, define o tamanho do que vai ser gravado) em rt

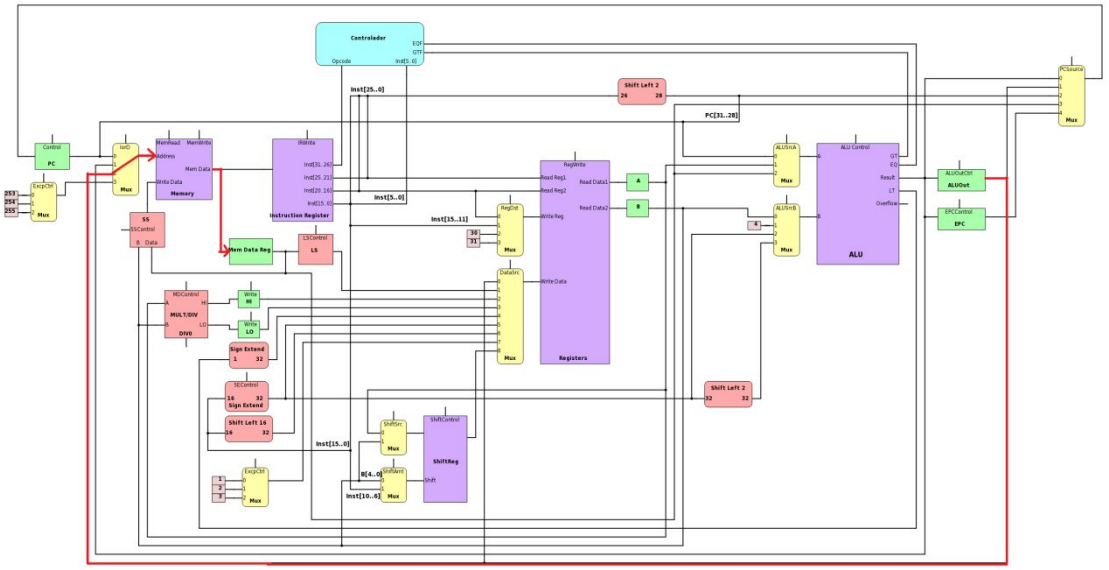


SW rt, offset(rs)
SH rt, offset(rs)
SB rt, offset(rs)

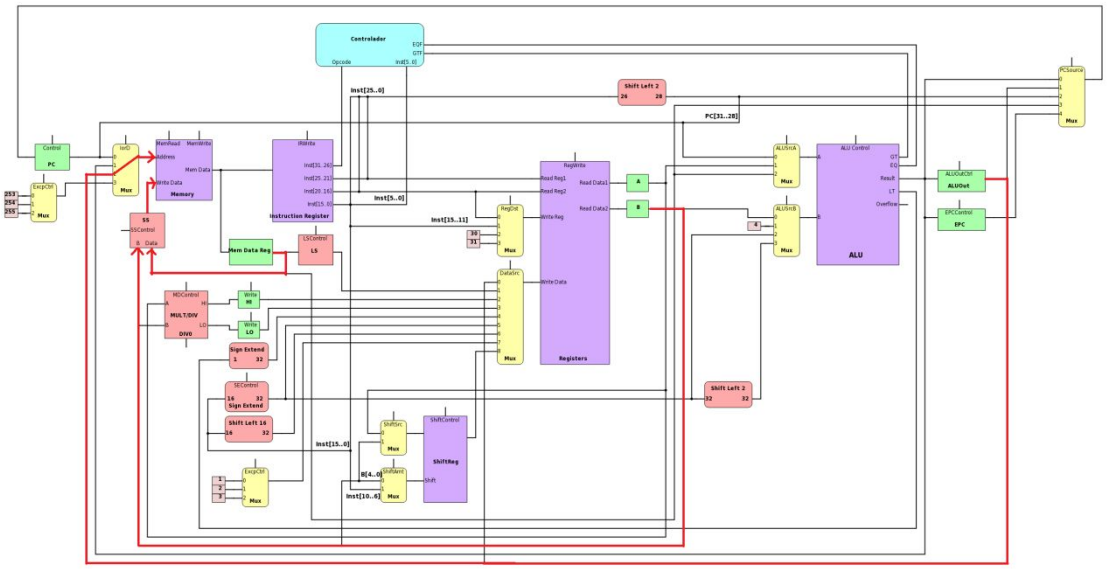
3.Calcula a soma de rs com ‘offset’ estendido para 32 bits, e salva em ALUOut



4.Carrega a informação contida na posição de memoria definida por ALUOut em MDR

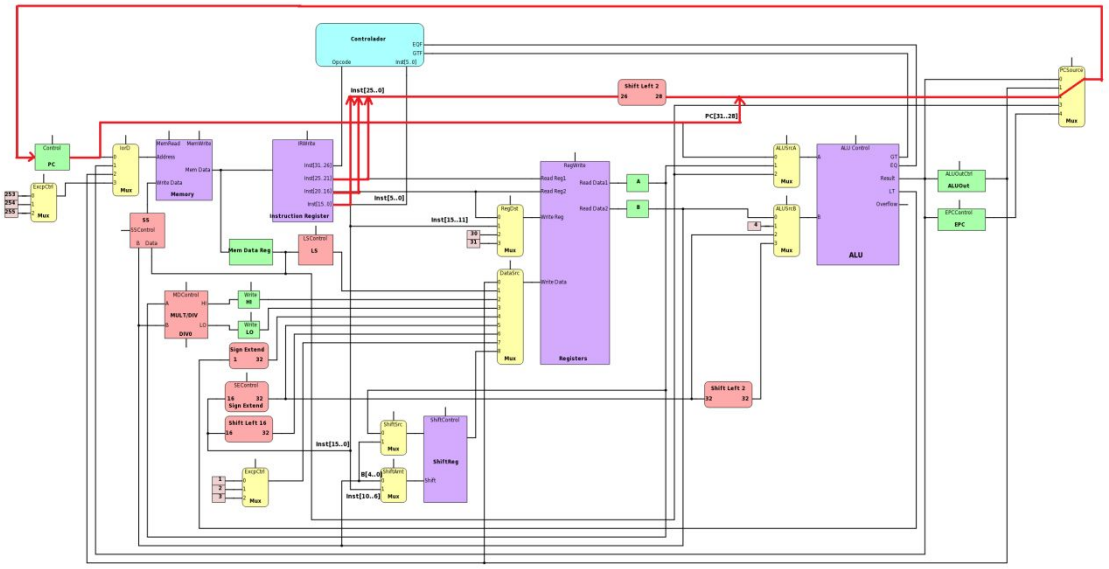


5.Grava o valor do registrador B na posição de memória definida por ALUOut, após passar por SS(Store Size, define o tamanho do que vai ser gravado, sem perda de informação)



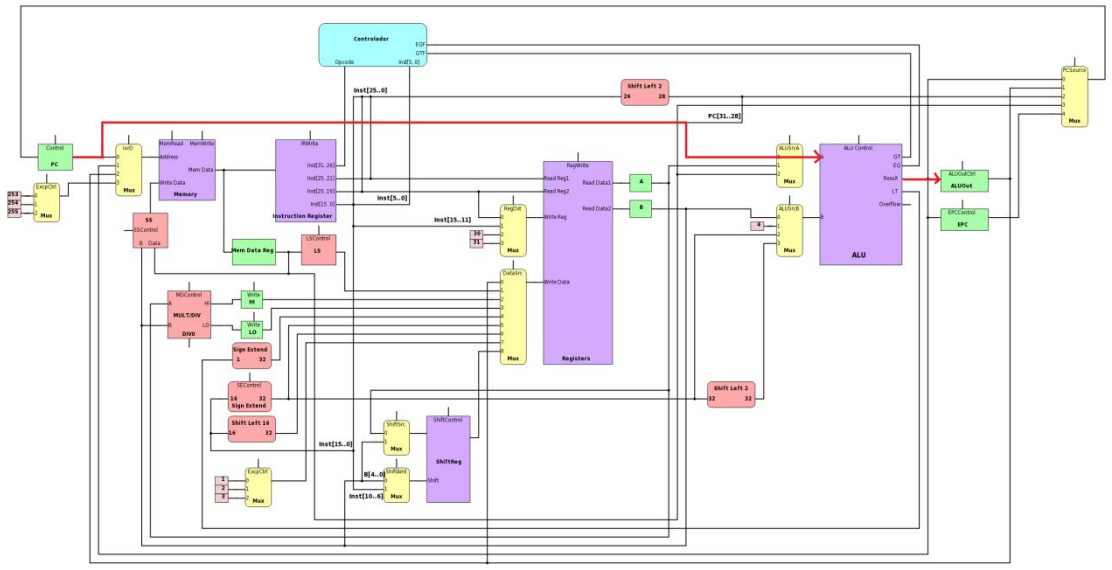
J offset

3.Desloca ‘offset’ em 2 bits para a esquerda, concatena com [31..28] de PC e grava o resultado em PC



JAL offset

3.Grava o valor de PC+4 em ALUOut



4. Desloca ‘offset’ em 2 bits para a esquerda, concatena com [31..28] de PC, grava o resultado em PC e grava o valor de ALUOut no registrador 31