

Universidade Federal do Rio Grande do Sul
Instituto de Informática



INF01113
Organização de Computadores B

Trabalho Prático 1 - Tarefa 09

Implementação de Instruções no MIPS

Bruno Alexandre Hofstetter Bourscheid (00550177)
Fernando Longhi de Andrade (00580366)
Luiz Augusto Ponzoni Schmidt (00580108)
Miguel Dutra Fontes Guerra (00342573)
Pedro Lubaszewski Lima (00341810)

Turma B

19 de abril de 2025

Sumário

1.1	MIPS Singlecycle	2
1.1.1	Modificações no Bloco Operativo	2
1.1.2	Modificações no Bloco de Controle	3
1.1.3	Testes de Implementação	5
2.1	MIPS Multicycle	8
2.1.1	Modificações no Bloco Operativo	8
2.1.2	Modificações no Bloco de Controle	8
2.1.3	Estados de Controle	8
2.1.4	Testes de Implementação	8
3.1	MIPS Pipeline	9
3.1.1	Modificações no Bloco Operativo	9
3.1.2	Modificações no Bloco de Controle	10
3.1.3	Testes de Implementação	11

1.1 MIPS Singlecycle

1.1.1 Modificações no Bloco Operativo

Para começar, segue uma ilustração de como estava inicialmente a parte operativa do MIPS Singlecycle:

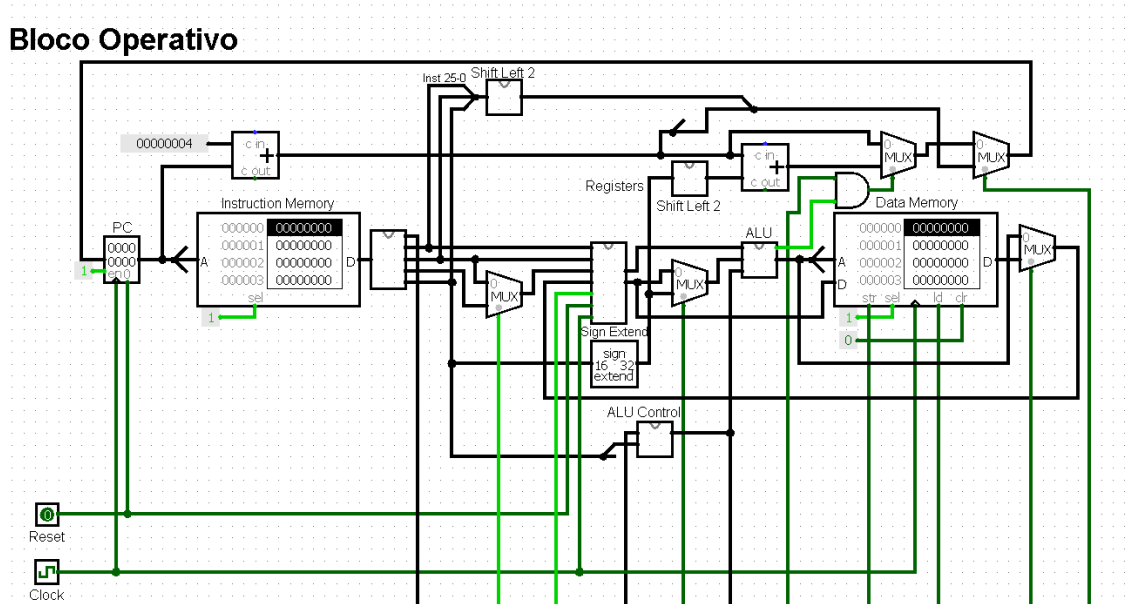


Figura 1: MIPS Singlecycle - Bloco Operativo Antes

Essa parte operativa foi modificada de acordo com as instruções adicionadas ao MIPS Singlecycle:

- Adicionada operação multiplicação na ALU;
- Adicionada saída dos bits superiores da multiplicação na ALU;
- Adicionada saída IsMULT da ALU;
- Adicionados registradores especiais LO e HI;
- Atualizada a flag 0 na ALU;
- Adicionado comparador para BLTZ e SLTIU;
- Adicionado sign extend para o BLTZ e SLTIU;
- Adicionado splitter para LB;
- Adicionado sign extend para LB;
- Atualizado o splitter para a Data Memory.

Segue uma ilustração de como ficou a parte operativa do MIPS Singlecycle após as modificações acima listadas:

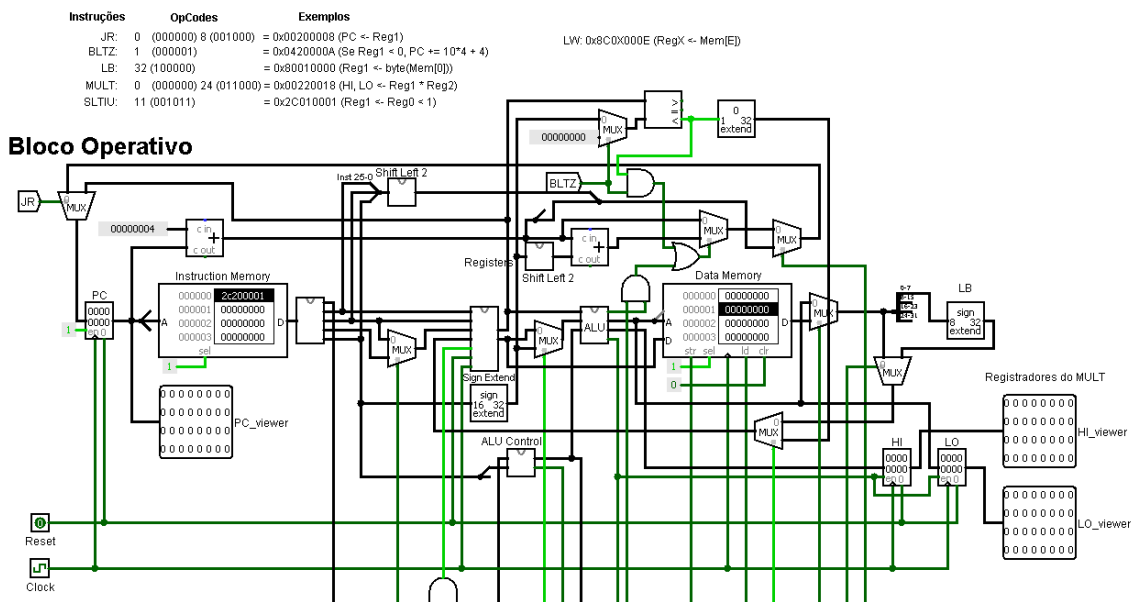


Figura 2: MIPS Singlecycle - Bloco Operativo Depois

1.1.2 Modificações no Bloco de Controle

Já para a parte de controle, segue uma representação inicial do MIPS Singlecycle:

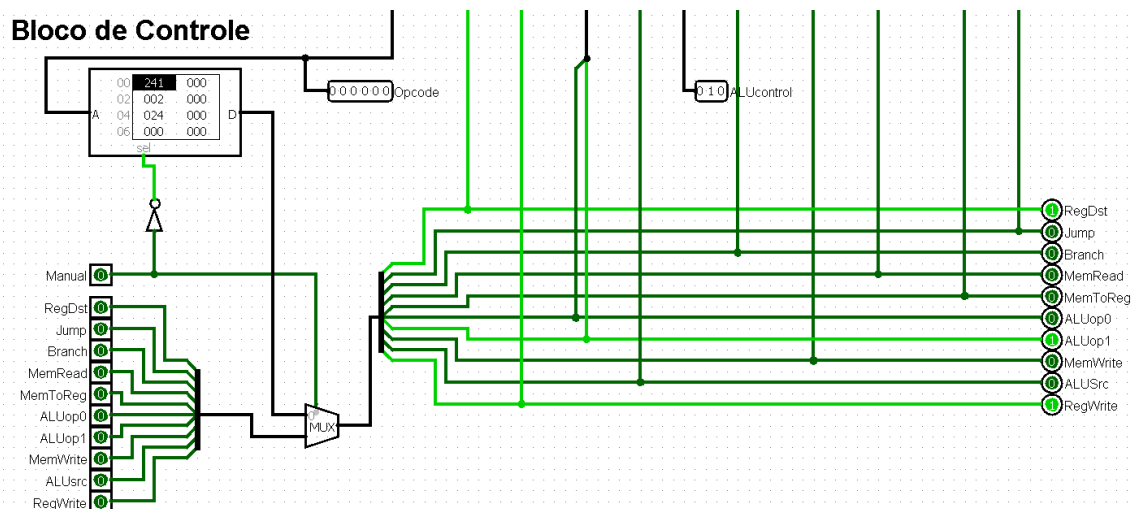


Figura 3: MIPS Singlecycle - Bloco Controle Antes

Com ela em mente, foram feitas as seguintes modificações:

- Criados os sinais de controle LB, SLTIU, BLTZ, JR e IsMult;
- Atualizado o ALU Control para as instruções de MULT e JR;
- Adicionado MUX para o BLTZ;
- Adicionado controle de branch para BLTZ;
- Adicionado controle de RegWrite usando IsMULT;
- Adicionado MUX para o JR;
- Adicionado MUX para LB;

- Abaixo segue também a codificação em memória ROM de controle das novas instruções:
- ROM para JR: 0x0241 end 0x00;
 - ROM para BLTZ: 0x0404 end 0x01;
 - ROM para LB: 0x0B18 end 0x20;
 - ROM para MULT: 0x0241 end 0x00;
 - ROM para SLTIU: 0x1300 end 0x0B.

The diagram illustrates the internal logic of the Control Block (Bloco de Controle). It features a 3-to-8 decoder (000 to 111) with inputs A, B, and C. The decoder's outputs are connected to various control signals, including RegDst, Jump, Branch, MemRead, MemToReg, ALUOp0, ALUOp1, MemWrite, ALUSrc, RegWrite, BLTZ, and JR. A 2-to-1 multiplexer (MUX) selects between the Manual input and the decoder's output for the RegWrite signal. The ALUControl signal is also shown, connected to the ALUOp0 and ALUOp1 inputs. The diagram shows the internal wiring and logic gates (AND, OR, NOT) that implement the control logic.

Algumas dessas modificações aqui listadas foram mostradas em mais detalhes na subseção anterior.

1.1.3 Testes de Implementação

Para comprovar que essas mudanças acima trouxeram a correta implementação das instruções especificados neste trabalho, abaixo seguem os testes realizados com imagens e a respectiva sequência de instruções:

- Teste da Instrução JR:

JR: $PC \leftarrow Reg1$

0x8C010000; $Reg1 \leftarrow Mem[0]$ (3)

0x00200008; $PC \leftarrow Reg1$ (3)

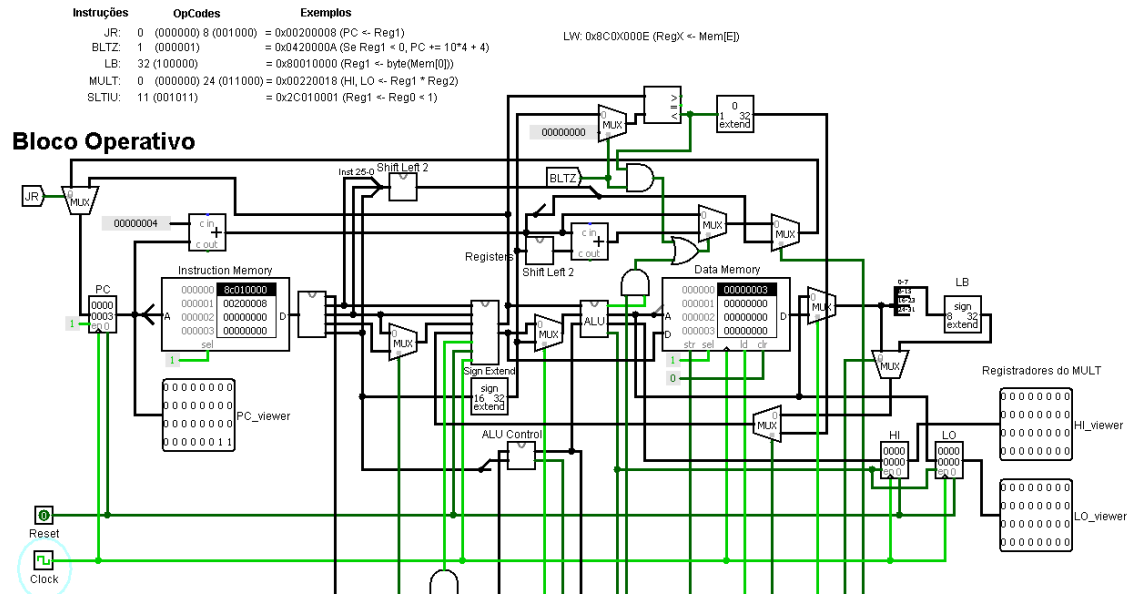


Figura 5: MIPS Singlecycle - Teste JR

- Teste da Instrução BLTZ:

BLTZ: if $Reg1 < 0$ then $PC += IMED \cdot 4 + 4$ ($IMED = 10$)

0x8C010000; $Reg1 \leftarrow Mem[0]$ (-1 (0xFFFFFFFF))

0x0420000A; $PC += IMED \cdot 4 + 4$ ($PC = 48$)

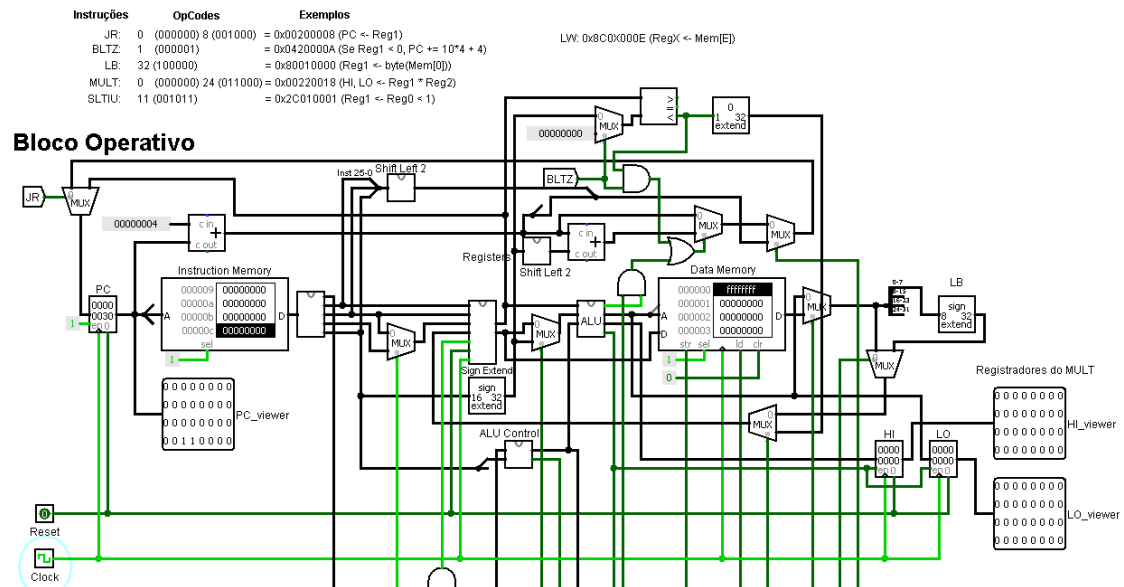
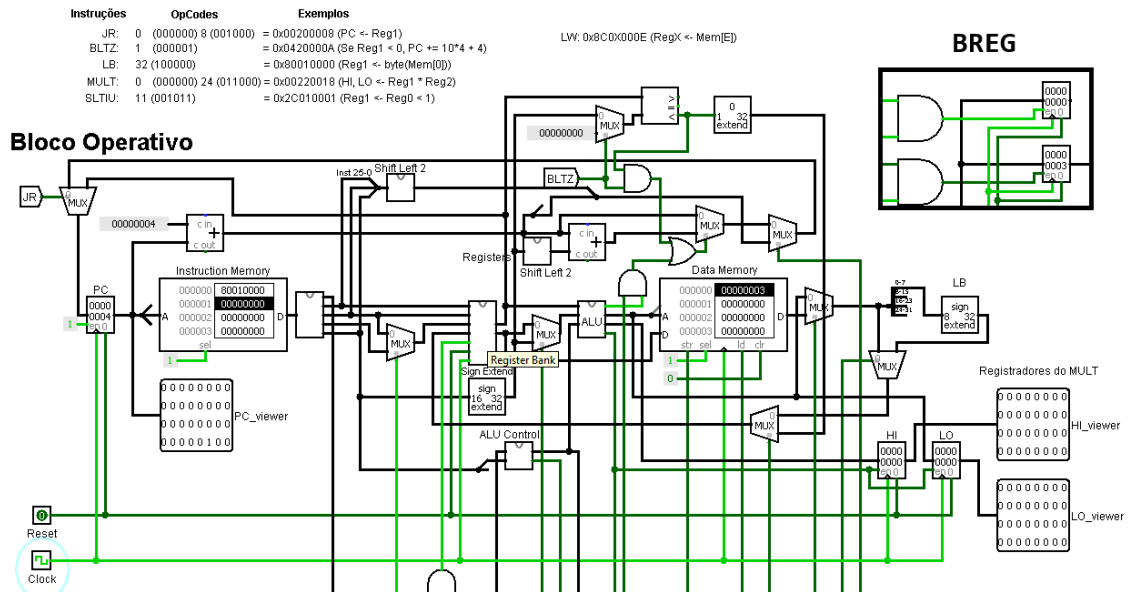
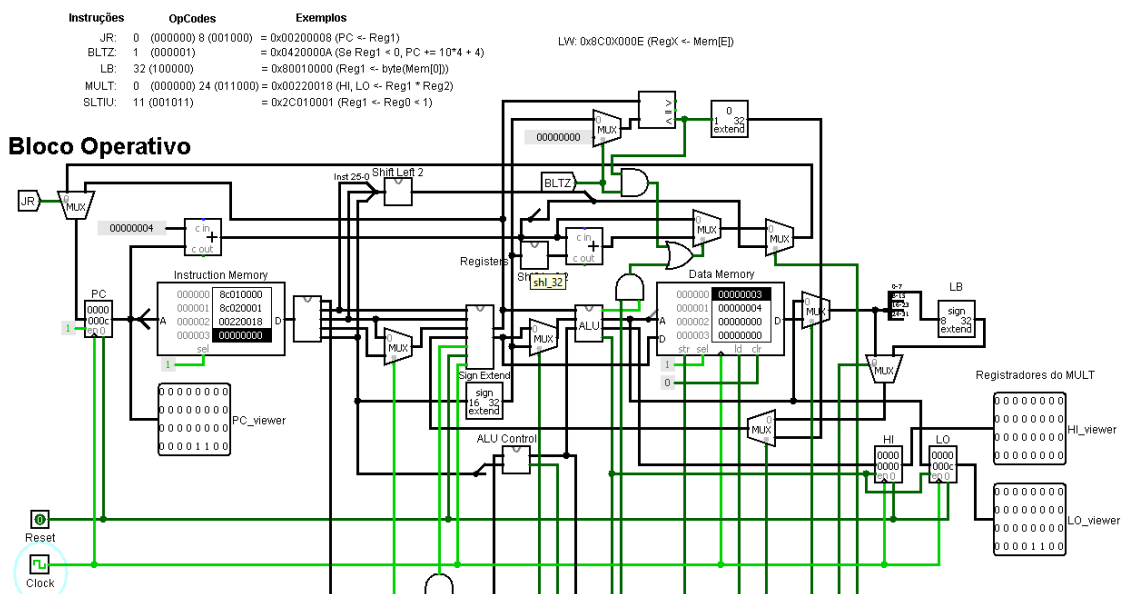


Figura 6: MIPS Singlecycle - Teste BLTZ

- LB: Reg1 \leftarrow byte(Mem[0]) (3)
 0x80010000; Reg1 \leftarrow byte(Mem[0])



- ```
MULT: HI, LO ← Reg1 · Reg2
0x8C010000; Reg1 ← Mem[0] (3)
0x8C020001; Reg2 ← Mem[1] (4)
0x00220018; HI, LO ← Reg1 · Reg2 (HI = 0, LO = 12)
```



- Teste da Instrução SLTIU:

SLTIU:  $\text{Reg1} \leftarrow \text{Reg0} < \text{IMED} (1)$

0x2C010001;  $\text{Reg1} \leftarrow \text{Reg0} < 1$  ( $\text{Reg1} = 1$ )

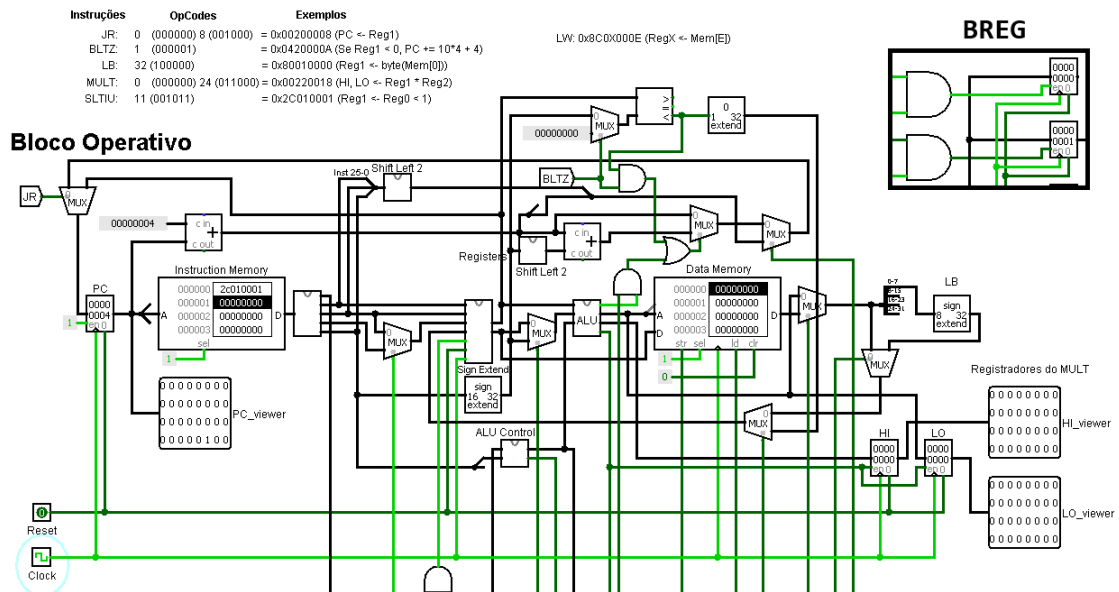


Figura 9: MIPS Singlecycle - Teste SLTIU



## 2.1 MIPS Multicycle

### 2.1.1 Modificações no Bloco Operativo

### 2.1.2 Modificações no Bloco de Controle

### 2.1.3 Estados de Controle

São utilizados no projeto 12 estados diferentes, contendo instruções de 3 a 5 ciclos, todas representadas na FSM anexada a seguir: Nota-se que os nodos representam os estados de controle,

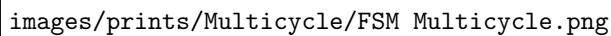
The image is a placeholder for a diagram titled "FSM Multicycle.png". It is located within a large rectangular frame. The text "images/prints/Multicycle/FSM Multicycle.png" is visible within the frame, indicating the location of the image file.

Figura 10: MIPS Multicycle - Máquina de estados Multicycle

enquanto as arestas os OPCODEs, assim formando uma máquina de Moore com as saídas em "1" de cada estado notadas em cada nodo.

### 2.1.4 Testes de Implementação

## 3.1 MIPS Pipeline

### 3.1.1 Modificações no Bloco Operativo

Primeiramente, segue uma visão de como estava a implementação do MIPS com pipeline antes do acréscimo das novas instruções:

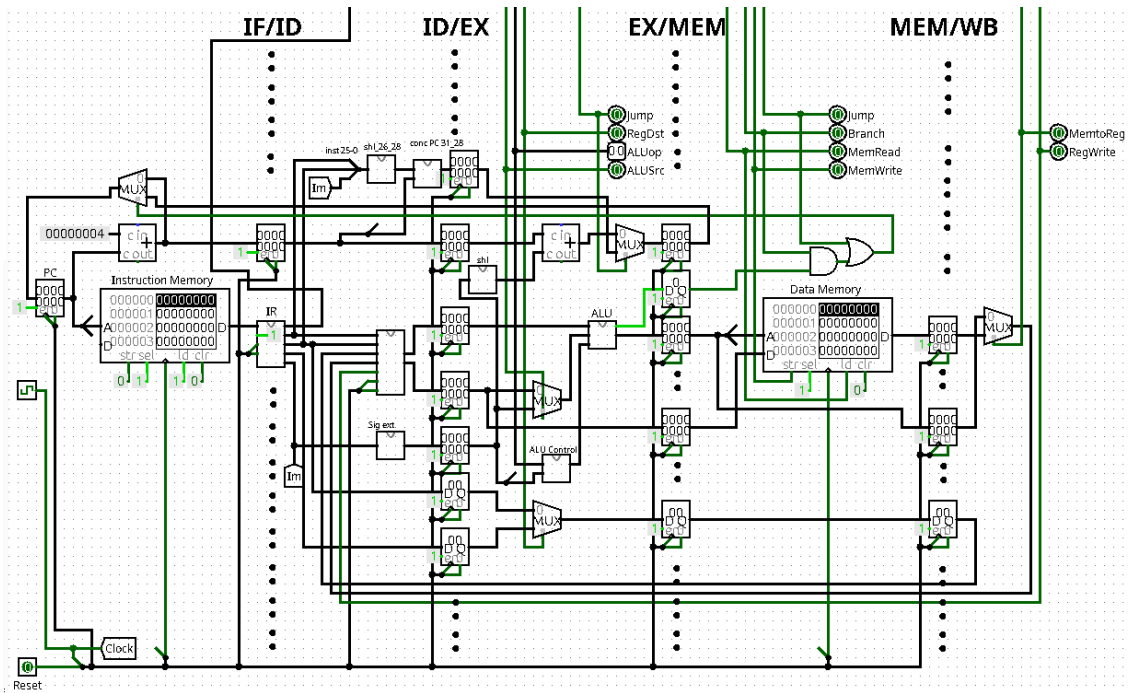


Figura 11: MIPS Pipeline - Bloco Operativo Antes

Sobre essa versão do MIPS com pipeline, foram feitas as modificações listadas à seguir:

- Adicionada operação multiplicação na ALU;
- Adicionada saída dos bits superiores da multiplicação na ALU;
- Adicionada barreira temporal para capturar valor de RS;
- Adicionado MUX para o JR na entrada do PC;
- Corrigidos os timings entre barreiras temporais para o sinal de IsMult;
- Utilizado o sinal IsMult para não habilitar a escrita nos registradores;
- Propagado o sinal de imediato com 32 bits para a etapa de MEM;
- Colocado comparador de menor que 0 ou menor que RS para instruções de BLTZ e SLTIU em MEM;
- Conectado o BLTZ ao mux de branches;
- Propagado o bit de comparação RS ; Imed para o estágio de MEM;
- Adicionado bit extender para o resultado da comparação acima;
- Conectado o resultado desse bit extender na entrada de escrita dos GPRs;
- Adicionado splitter para pegar os 8 bits menos significativos de um dado da memória;
- Extendido esses 8 bits acima;
- Conectado esse resultado controlado pelo sinal LB nos GPRs.

Com essas modificações listadas acima, obteve-se a seguinte nova representação do bloco operativo do MIPS:

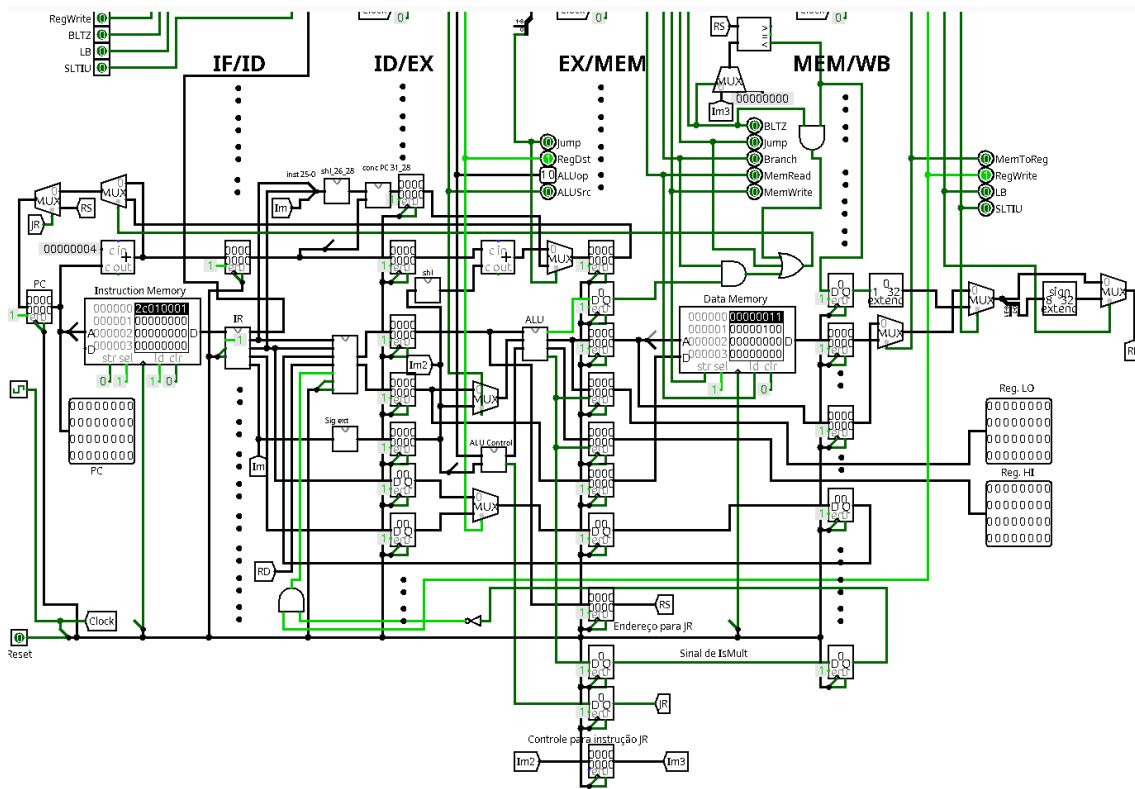


Figura 12: MIPS Pipeline - Bloco Operativo Depois

### 3.1.2 Modificações no Bloco de Controle

Já na parte de controle, anteriormente, tinha-se a seguinte representação:

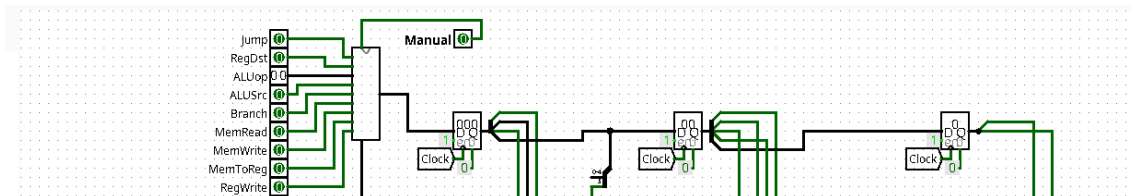


Figura 13: MIPS Pipeline - Bloco Controle Antes

A partir dessa parte de controle, é possível listar as seguintes modificações:

- Criados os sinais de controle LB, SLTIU, BLTZ, JR e IsMult;
- Atualizado o ALU Control para as instruções de MULT e JR;
- Adicionada saída IsMULT da ALU;
- Adicionada barreira temporal para o estágio de EX/MEM do JR;
- Propagados todos os sinais de controle para as respectivas etapas do circuito:
  - BLTZ em MEM;
  - LB e SLTIU em WB.

Além dessas modificações, foram reordenados os sinais de controle para serem equivalentes aos sinais de controle do MIPS Singlecycle. Assim, as modificações da ROM de controle foram as mesmas listadas na seção do MIPS Singlecycle:

- ROM para JR: 0x0241 end 0x00;
- ROM para BLTZ: 0x0404 end 0x01;
- ROM para LB: 0x0B18 end 0x20;

- ROM para MULT: 0x0241 end 0x00;
- ROM para SLTIU: 0x1300 end 0x0B.

Com tudo isso, obteve-se o seguinte novo esquemático externo da parte de controle do MIPS com pipeline:

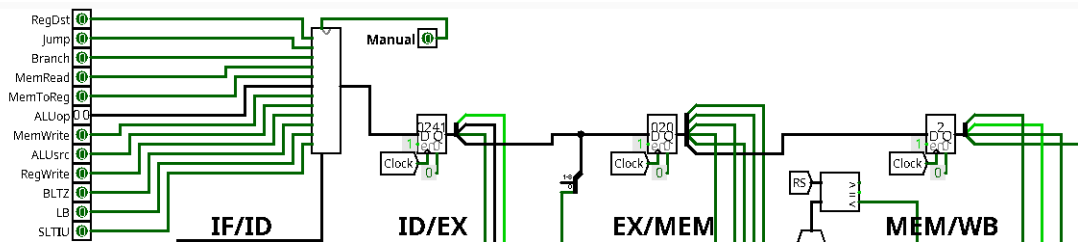


Figura 14: MIPS Pipeline - Bloco Controle Depois

Algumas dessas modificações aqui listadas foram mostradas em mais detalhes na subseção anterior.

### 3.1.3 Testes de Implementação

Para comprovar que essas mudanças acima trouxeram a correta implementação das instruções especificados neste trabalho, abaixo seguem os testes realizados com imagens e a respectiva sequência de instruções:

É importante notar, em todos os testes, que há uma grande quantidade de NOPs espalhados entre as instruções. Isso é devido a não terem sido implementadas soluções para lidar com dependências de dados e de controle.

- Teste da Instrução JR:

0x8C010000; LW: Reg1  $\leftarrow$  Mem[0] (64 ou 0x40)

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x00200008; JR: PC  $\leftarrow$  Reg1 (64 ou 0x40)

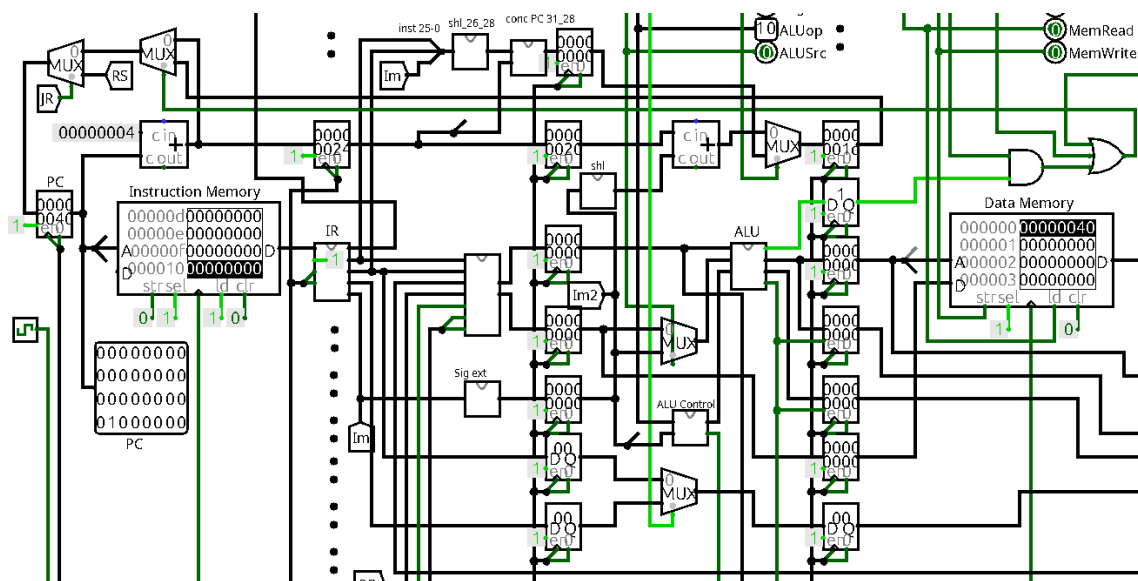


Figura 15: MIPS Pipeline - Teste JR

- Teste da Instrução MULT:

MULT: HI, LO  $\leftarrow$  Reg1  $\cdot$  Reg2

0x8C010000; Reg1  $\leftarrow$  Mem[0] (17 ou 0x11)

0x8C020001; Reg2  $\leftarrow$  Mem[1] (256 ou 0x100)

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x00220018; HI, LO  $\leftarrow$  Reg1  $\cdot$  Reg2 (HI = 0, LO = 4352 ou 0x10400)

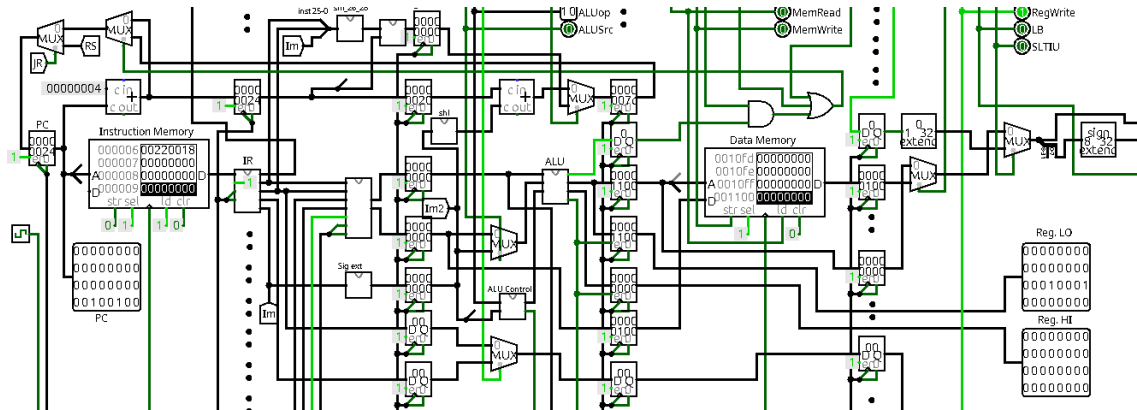


Figura 16: MIPS Pipeline - Teste MULT

- Teste da Instrução BLTZ:

BLTZ: if Reg1 < 0 then PC += IMED·4 + 4 (IMED = 10)

0x8C010000; Reg1 ← Mem[0] (-1 (0xFFFFFFFF))

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x00000000; NOP

0x0420000A; PC += IMED·4 + 4 (PC = 64 ou 0x40)

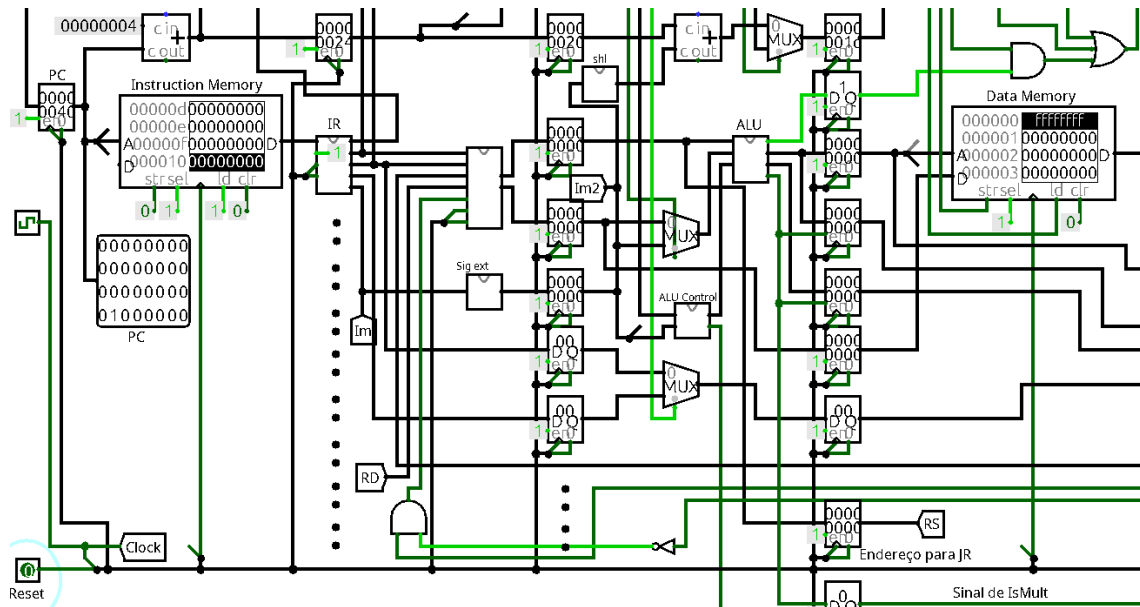


Figura 17: MIPS Pipeline - Teste BLTZ

- Teste da Instrução LB:

LB: Reg1 ← byte(Mem[0]) (0x000000FF)

0x80010000; Reg1 ← byte(Mem[0]) (0xFFFFFFFF)

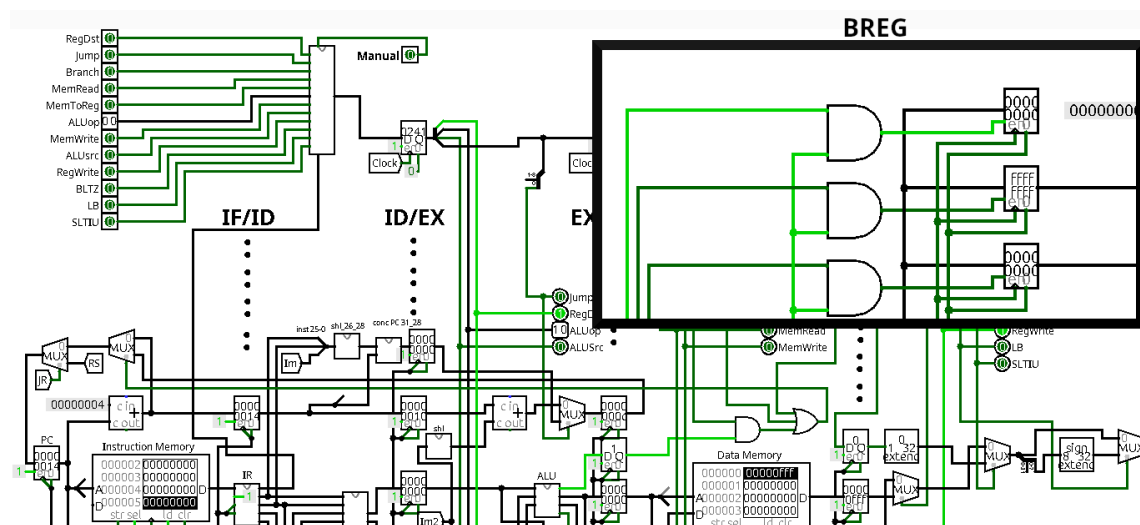


Figura 18: MIPS Pipeline - Teste LB

- Teste da Instrução SLTIU:

SLTIU:  $\text{Reg1} \leftarrow \text{Reg0} < \text{IMED} (1)$

0x2C010001;  $\text{Reg1} \leftarrow \text{Reg0} < 1$  ( $\text{Reg1} = 1$ )

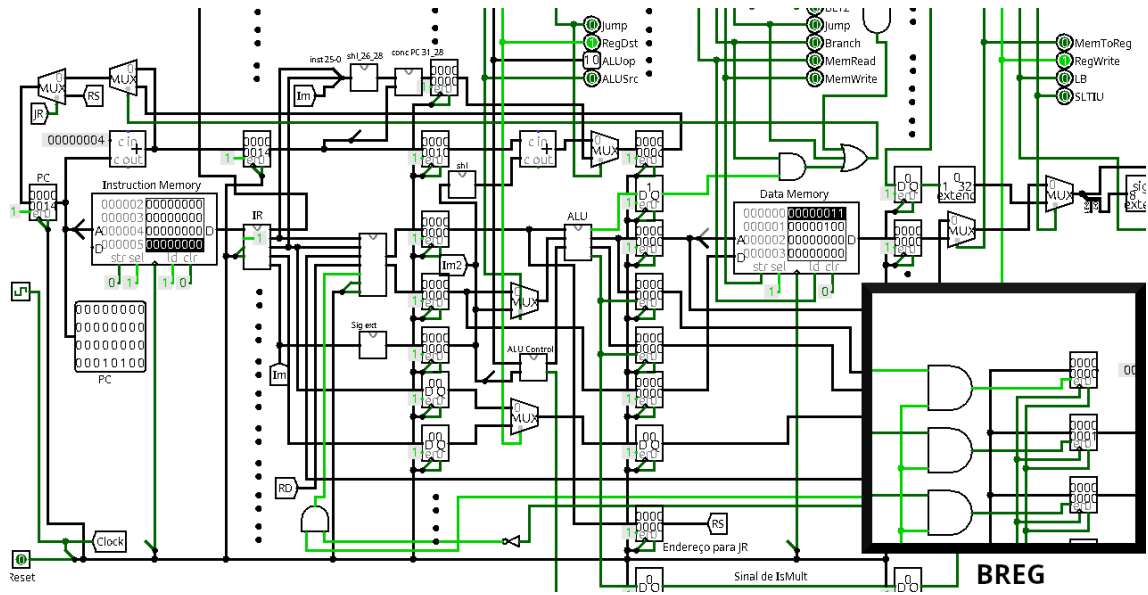


Figura 19: MIPS Pipeline - Teste SLTIU