| instruções  —-------->  \_\_\_\_\_\_\_\_\_  sinais de controle  |  \/ | L  O  A  D | S  T  O  R  E | A  D D | A D D I | J  U  M  P | S  E  T  Z | S  L  T | B  E  Q  Z | H  A  L  T | I  N  V  S  I  N | I  N  V  T  U  D | J  U  M  P  R  E  G |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EscrMem | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| LerMem | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Jump | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Halt | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| UlaOP | XX | X | 00 | 00 | X | 10 | 01 | 00 | X | X | X | X |
| BeqZ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| EscreReg | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| FonteULA | XX | X | 1 | 0 | X | 0 | 10 | 0 | X | X | X | X |

**ULA OP**

| Operação → | SOMAR | >(ativar se menor que) | AND |  |
| --- | --- | --- | --- | --- |
| Codigo → | 00 | 01 | 10 | 11 |

**FONTE ULA**

| SELECAO→ | 0 | 1 | 2 |  |
| --- | --- | --- | --- | --- |
| Codigo → | 00 | 01 | 10 | 11 |

**Observação:**

O passo inicial de todas as instruções é ler o endereço de PC e decodificar a instrução, por isso esse trecho será omitido da descrição do caminho de dados das instruções.

**Load:**

Os bits de 4 a 2 indicam o registrador que contém o endereço de onde o dado será buscado, já os bits 1 e 0 são concatenados com um zero à esquerda e assim indicam em qual registrador o dado buscado será gravado. O Mux que define o registrador a ser escrito seleciona o registrador indicado pelos bits 0 e 1 da instrução, o endereço é passado para a memória de dados, que faz a leitura do dado. O primeiro Mux que seleciona o que será gravado no registrador destino seleciona 0, o segundo e o terceiro também, assim o banco de registradores recebe o dado e o grava no registrador destino.

**Store:**

Funciona de maneira similar ao Load, porém desta vez os bits 1 e 0 da instrução representam o registrador que armazena o conteúdo que será gravado na memória. A memória recebe o endereço e o conteúdo, realizando assim a gravação.

**Add:**

Os bits de 4 a 2 indicam o primeiro registrador a ser lido, esses bits são incrementados em 1 e passam a indicar o registrador subsequente ao indicado na instrução, que também é lido pelo banco de registradores. Os bits 1 e 0 indicam o registrador que será gravado, por isso o mux seleciona 0. Os Mux superior da ULA seleciona 1, e consequentemente o Mux inferior também seleciona 1, dessa forma, o conteúdo do registrador indicado pelos bits 4 a 2 é somado com o conteúdo de seu subsequente, já que o mux seguinte ao superior da ULA também seleciona 0. O resultado da Ula é selecionado pelo Mux seguinte, que em sequência passa pelos próximos 2 Mux e é gravado no registrador destino.

**Addi:**

Os bits de 4 a 2 indicam o imediato que será somado com o conteúdo do registrador de destino, indicado pelos bits 1 e 0. O imediato passa por um extensor de sinal e passa a ter 8 bits. O Mux superior e inferior da Ula selecionam 1, desta forma o conteúdo do registrador 2 é somado com o imediato de sinal estendido. O Mux seguinte o superior da ULA seleciona 0. O resultado da operação que sai da ULA segue o mesmo caminho do resultado obtido através da instrução Add.

**Jump:**

Os bits de 4 a 2 indicam o offset do jump, que tem seu sinal estendido e é somado ao valor de PC+1. O mux anterior de PC seleciona 0 e o que vem em seguida seleciona 1, dessa maneira escrevendo em PC o resultado do Offset do jump + PC+1.

**Set to zero:**

Os bits de 4 a 2 representam o registrador que será zerado, assim, o Mux determina que o registrador a ser escrito é o representado por estes bits. Os registradores que podem ser informados são t1 a t4, ou seja, de (000) a (011), assim quando estendemos o sinal, dos bits de 4 a 2, garantidamente teremos um número de 8 bits cujo bit mais significativo [7] é 0. Este valor obtido é passado para a ULA, como o Mux superior e inferior selecionam 0 e o seguinte ao inferior também seleciona 0, a ULA recebe o valor cujo sinal foi estendido e um “valor lixo” lido de um registrador representado pelos dois últimos bits da instrução que necessariamente são X1. Ou seja, o valor lido será do registrador 3 ou 4, o que não altera em nada, visto que ao ser enviado a ULA juntamente com o valor de sinal estendido e for realizada a operação AND, garantidamente o bit mais significativo [7] será 0. O Mux seleciona o resultado da ULA, que ao passar pelo próximo Mux seleciona 1, desta maneira pegando apenas o bit mais significativo do resultado e concatena com 7 zeros à esquerda. O Mux seguinte seleciona 0 e assim o valor zero será gravado no registrador destino.

**Set on Less than:**

Os bits de 4 a 2 representam o primeiro registrador a ser lido e os bits 1 e 0 representam tanto o segundo registrador a ser lido quanto o registrador em que será gravado o resultado. Os Mux da Ula selecionam 2, enviando assim ambos os dados lidos para a ULA, que tem como resultado 1 em caso positivo e 0 em caso negativo. O resultado da Ula é selecionado pelo MUX. Os dois Mux seguintes selecionam 0 e assim o resultado é gravado no registrador de destino.

**Branch if Equal Zero:**

Os bits de 4 a 2 representam o Offset que será somado a PC+1 em caso de salto, os bits 1 e 0 representam o registrador cujo conteúdo será comparado a 0. O segundo MUX superior da ULA seleciona 1, enviando assim 0 para a ULA, já o MUX inferior seleciona 0 e envia o conteúdo do registrador para a ULA, que faz a comparação, realizando uma soma ou subtração, por exemplo, o que realmente importa é a saída 0. A saída zero é enviada para uma porta AND com o sinal de controle BEQZ, em caso positivo, isto seleciona 1 no Mux posterior de PC. Paralelamente, o Offset tem seu sinal estendido e é somado com PC+1, o resultado dessa soma passa pelo Mux anterior de PC, que seleciona 0, levando ao Mux posterior, que, como já explicado, seleciona 1 e escreve em PC o resultado desta soma.

**Halt:**

Primeiramente os bits mais significativos [7:5] e os bits menos significativo[1:0] entram no controle zerando todas as flags de controle, exceto pela do halt que será ativa e passará por uma porta NOT fazendo PC somar com zero e escrevendo novamente em PC a mesma instrução, assim, travando o sistema.

**Inverte sinal (ninvs):**

Os bits [4:2] serão lidos pelo banco de registradores selecionando o registrador alvo, o sinal de controle para escolher qual registrador será escrito é selecionado pelo bit mais significativo, assim escolhendo o mesmo registrador que está nos bits [4:2]. O conteúdo do registrador lido não passa pela ULA sendo direcionado diretamente para o somador que irá somar 10000000, entrando na porta 1 do mux onde o bit menos significativo da instrução selecionará a entrada 1. O Mux seguinte também seleciona 1, gravando assim este resultado no registrador destino.

**Inverte todos os bits (ninvt):**

Semelhante à instrução anterior, com a diferença de que os bits lidos passam por portas NOT ao invés de serem somados. O primeiro Mux que antes selecionava 1, agora seleciona 0, que passa os bits negados para o mux seguinte, que ainda seleciona 1 e grava o resultado no registrador destino.

**Jump para o registrador (njre):**

Os bits de 4 a 2 representam o registrador que contém o Offset que será somado com PC+1. O banco de registrador lê o valor, que é somado com PC+1 , o Mux anterior e posterior de PC selecionam 1, gravando assim em PC o valor calculado.

**Apêndice**

Tabela de instruções atualizadas:

Em azul as instruções que sofreram alterações.

| OPcode (3bits) | mnemônico | parte 1 | parte 2 | parte 3 |
| --- | --- | --- | --- | --- |
| Load(000) 0 | nlw | RegEndereço(3bits) | regDestino(2bits) |  |
| Store(101)1 | nsw | RegEndereço(3bits) | regConteúdo(2bits) |  |
| Add(010)2 | nadd | RegEndereço(3bits) | regDestino(2bits) |  |
| addi(011)3 | naddi | imediato(3bits) com sinal -> -3 a 3 | regDestino(2bits) |  |
| jump(100)4 | nj | off set com sinal (3bits ) -4 a 4 | 1 bit irrelevante | (1 bit de verificação)  se for 0 é o jump |
| set to zero(100)4 | nstz | (3 bits) Registrador alvo. | 1 bit irrelevante | (1 bit de verificação)  se for 1 é o set to zero |
| set less than(001)5 | nslt | regConteudo(3bits) | regDestino(2bits) |  |
| beqz(110)6 | nbeqz | Offset(3bits)com sinal | endereço Conteúdo(2bits) |  |
| Halt(111)7 | nhalt | 3 bits irrelevante | (2bits de verificão)  se for 00 é Halt |  |
| inverter sinal(111)7 | ninvs | (3 bits) Registrador alvo. | (2bits de verificação)  se for 01 é inverter de sinal) |  |
| inverte tudo(111)7 | ninvt | (3 bits) Registrador alvo. | (2bits de verificação)  se for 10 é inverter todo os bits) |  |
| jump  reg(111)7 | njre | (3 bits) Registrador alvo. | (2bits de verificação)  se for 11 é inverter todo os bits) |  |

Tabela de comentário

| OPcode | Comentário |
| --- | --- |
| Load | Carrega uma palavra da memória para um registrador. |
| Store | Armazena o conteúdo de um registrador na memória. |
| Add | Dados dois registradores, soma o conteúdo do primeiro com o conteúdo do registrador subsequente, armazena o resultado no segundo registrador informado. |
| Addi | Soma uma constante ao conteúdo de um registrador.  Ex: soma o registrador destino com ele mesmo + um imediato. |
| Jump | Salta para a instrução indicada que é (PC+1 + OffSet). |
| Set to zero | Seta o registrador alvo para zero. |
| Set less than | Dados dois registradores, seta o segundo para 1 caso o conteúdo do primeiro registrador seja menor que o conteúdo do segundo, seta para 0 em caso contrário.  ex: t2 = 5, t3 = 10 -> t2<t3 => t3=1. |
| Beqz | Salta para o endereço correspondente a soma do Offset informado com PC+1, caso o conteúdo do registrador informado seja igual a 0. |
| Halt | Para o processo na linha em que foi chamado |
| Ninvs | Inverte o bit de sinal do registrador alvo |
| Ninvt | Inverte todos os bits do conteúdo do registrador. |
| Njre | Salta para o endereço correspondente a soma conteúdo do registrador informado com PC+1 |

Informações sobre os registradores:

| Registrador | Comentário |
| --- | --- |
| $t1(000)1 | Registrador aritmético 1 |
| $t2(001)2 | Registrador aritmético 2 |
| $t3(010)3 | Registrador aritmético 3 |
| $t4(011)4 | Registrador aritmético 4 |
| $zero(100)5 | Contém o valor zero |
| $pc(101)6 | Contém o endereço da linha +1 |
| $ra(110)7 | Registrador que pode ser usado para jr (a ser definido pelo programador, não é padrão) |
| $sa(111)8 | Fins diversos |