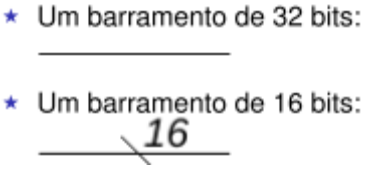
Construindo uma CPU - MIPS (parte 1)

Imprementação básica

- Vamos construir um processador MIPS básico, que implemente o seguinte
  - Instruções de referência de memória sw e lw
  - Instruções lógicas aritméticas tipo-R: add, sub, and, or e slt
  - Instruções lógicas e aritméticas tipo-I: addi, andi e ori
  - Instruções de desvio beq e j

Convenções iniciais

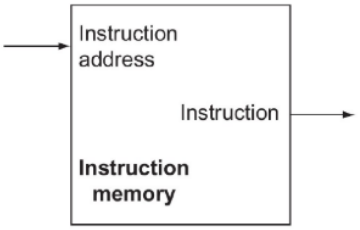
- O tamanho da palavra no MIPS é de 32 bits
- Assumiremos que as linhas de conexão no nosso projeto são na verdade um barramento (32 fios em paralelo) de 32 bits
  - Exemplos:



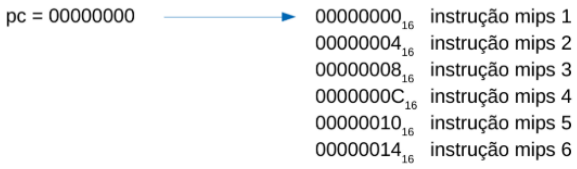
Caminho de dados

PRINCIPAIS COMPONENTES

- Uma **memória**, para armazenar as intruções
  - Entrada: o endereço da instrução
  - Saída: A instrução no endereço apontado



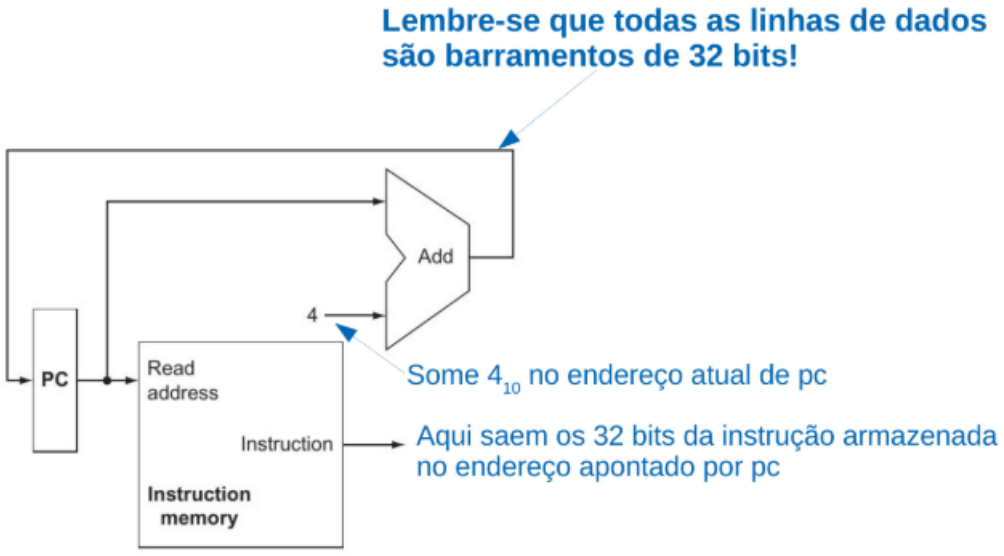
- **Registrador PC** no MIPS, para armazenar o endereço da próxima instução a ser executada
  - Entrada: O próximo endereço a ser armazenado no registrador
  - Saída: O endereço corrente



Incremento do PC

- Caso não tenhamos desvios, após a execução da instrução atual, devemos executar a próxima instrução
  - Precisamos adicionar mais 4 no Pc, por conta disso precisamos de um somador

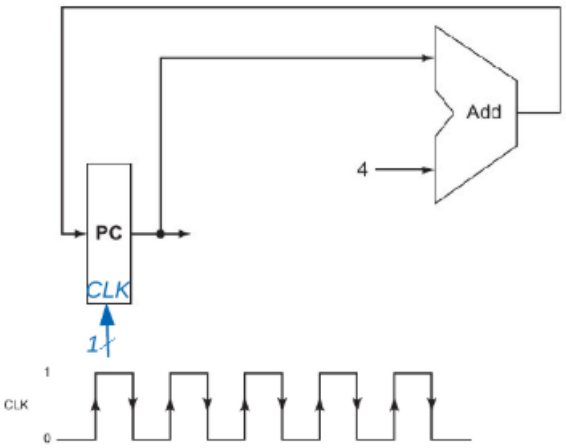
PC e memória de instrução



- O loop principal está pronto
  - Sempre enviamos a instrução para execução, e incrementamos pc em 4 para apontar para a próxima instrução

Sincornização

- Sinais de clock são adicionados nos elementos de estado (sequenciais)
  - O PC só vai ler a entrada na transição de clock
- Enquanto não há transição, o valor antigo do pc pe enviado para a saída e lido pelo adder



Instrução do tipo-R

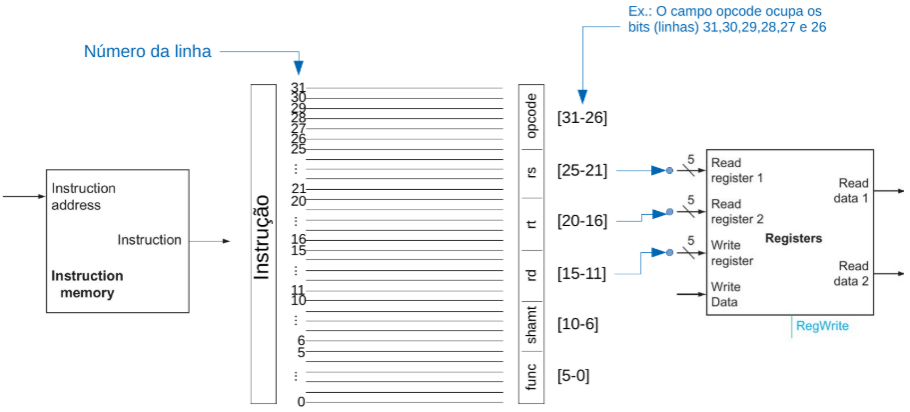
`add $a0, $a1, $a2`

- Leem dois registradores, executam uma operação aritmética em uma ALU, e armazenam o resultado em um terceiro registrador

BANCO DE REGISTRADORES

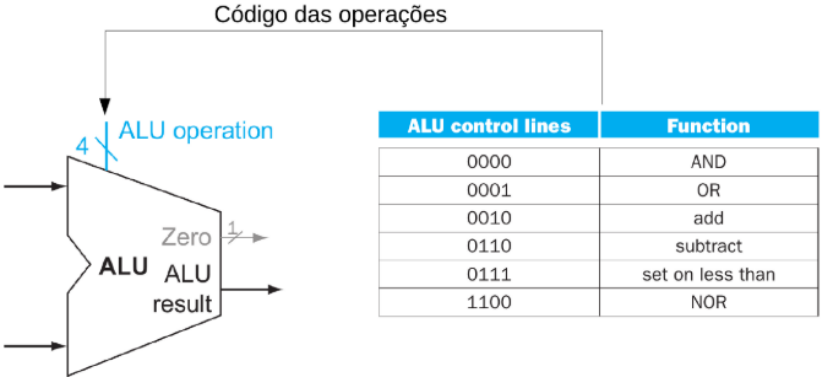
- Contém todos os 32 registradores do MIPS
- **Entradas**:
  - Endereço do registrador de leitura 1
  - Endereço do registrador de leitura 2
  - Endereço do registrador de escrita
  - Dados a serem escritos no registrador de escrita
- **Saídas:**
  - Dados do registrador de leitura 1 ȧ
  - Dados do registrador de leitura 2

FONTES DOS BITS



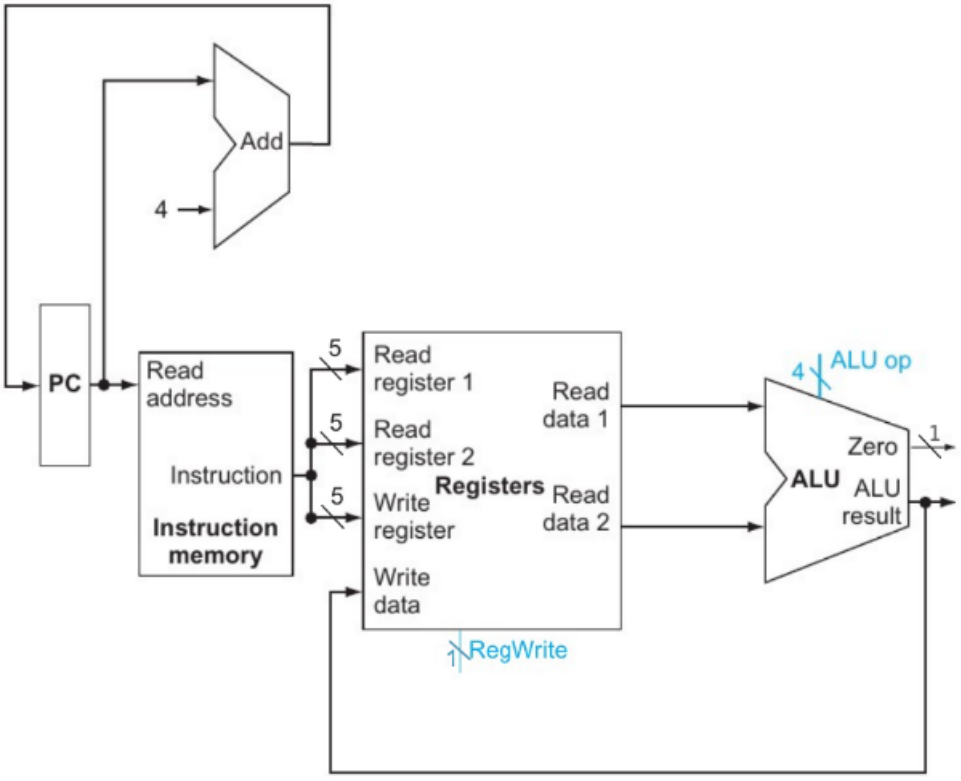
ALU

- As intruções precisam executar a operação com esses registradores, e o alu faz isso
- **Entradas:**
  - Dois valores de 32 bits ả
  - Uma entrada ALUop de 4 bits, que define qual a operação deve ser realizada com os valores
- Saídas:
  - Uma saída de 32 bits com o resultado da operação ả
  - Uma saída de 1 bit indicando se o resultado da operacão foi zero



ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

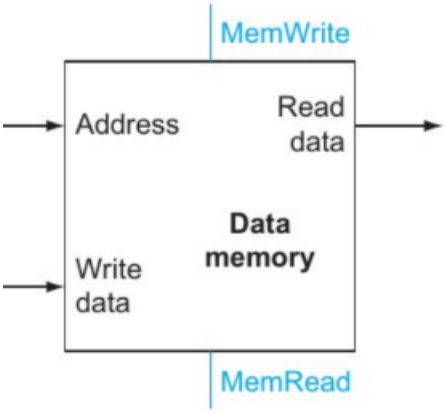
Ligando os componestes



Intruções do tipo-R (Loads e Stores)

- `lw` → $regDestino, deslocamento($regBase)
- `sw` → $regFonte, deslocamento($regBase)
- MEM[$regBase + deslocamento]

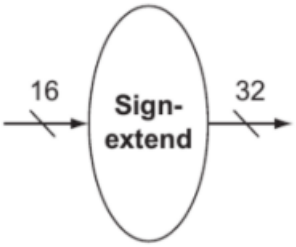
- Vamos precisar de uma memória para dados
- **Entradas:**
  - Endereço de memória ȧ
  - Dados a ser escrito ȧ(que é resgatado da memória ou que vai ser colocado na memória)
  - Sinais de Controle MemWrite e MemRead: Indica se a memória deve escrever na posição, ou se deve ler o dado especificado na posicão e direcioná-lo para a saída



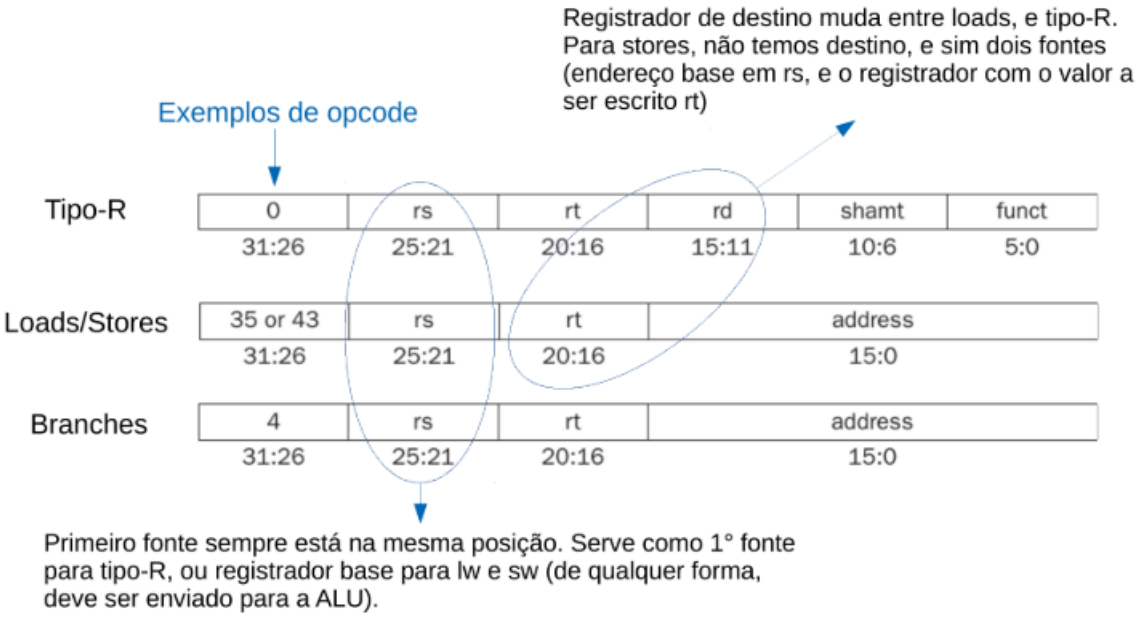
- O campo de contante tem 16 bits, e será somado com o registrador para obter o endereço de memória a ser lido/descrito, porem isso é um problema

EXTENSÃO DE SINAL EM COMPLEMENTO 2

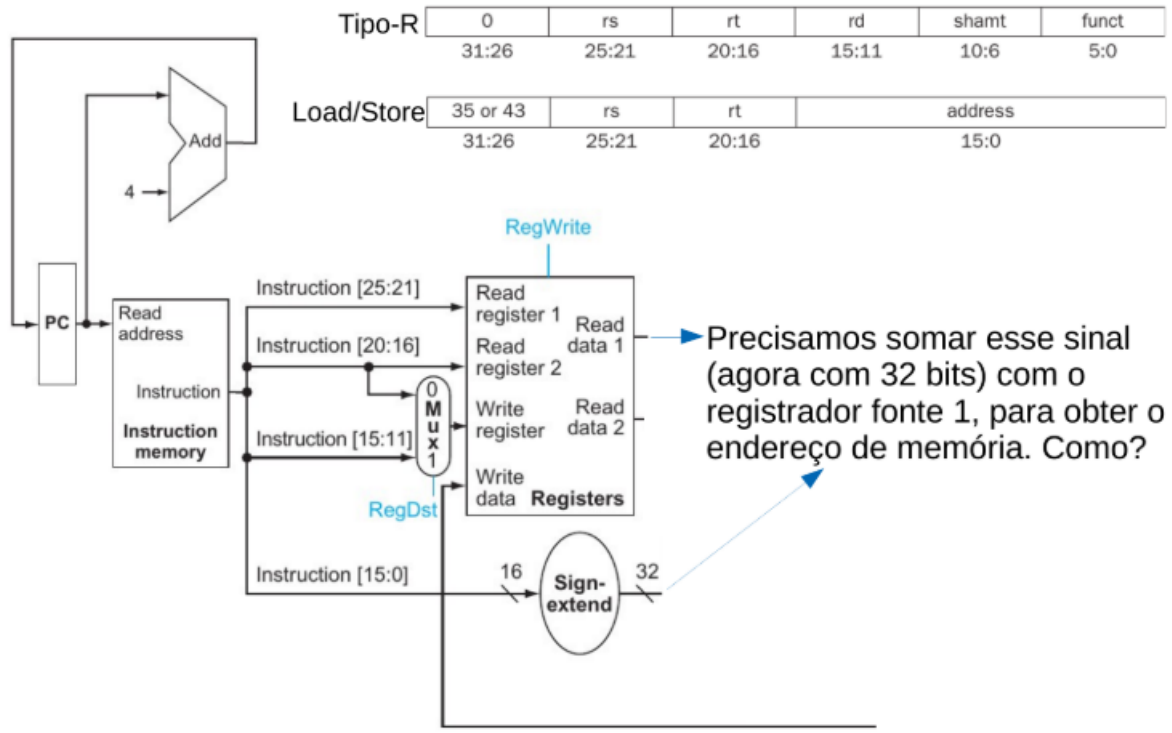
- Vamos utilizar um componente para extensão de sinal
  - Dado um sinal de 16 bits, gera o seu correspondente em 32 bits

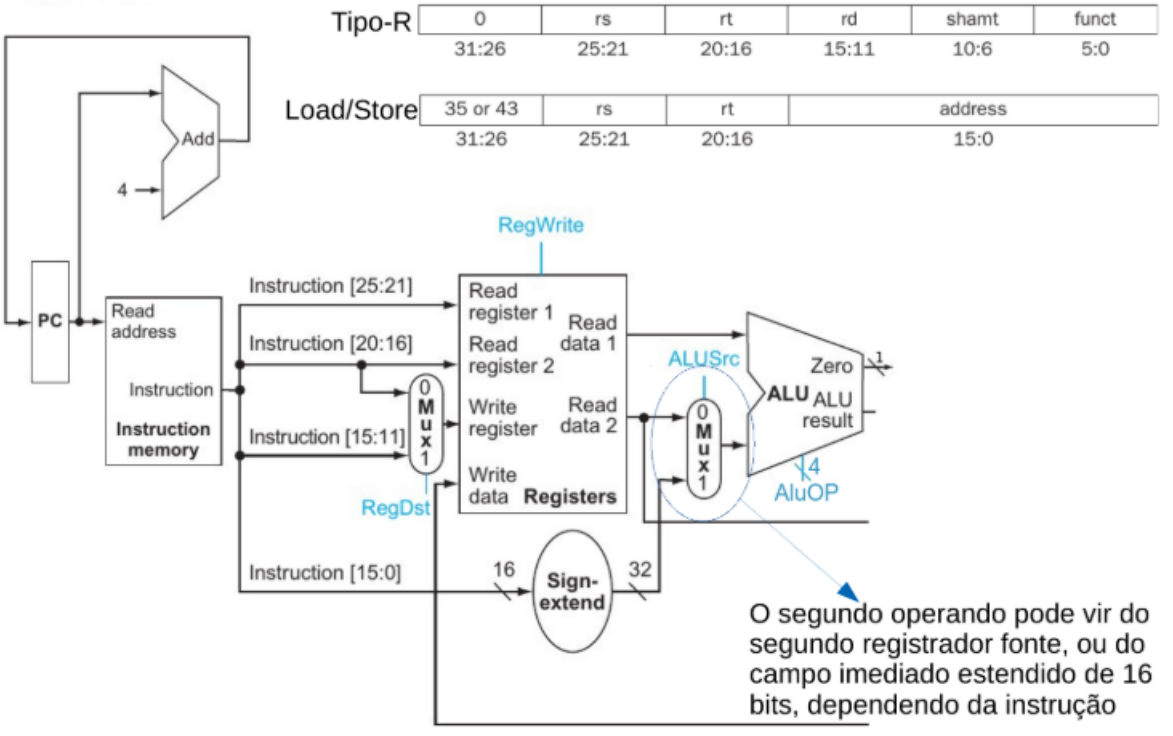


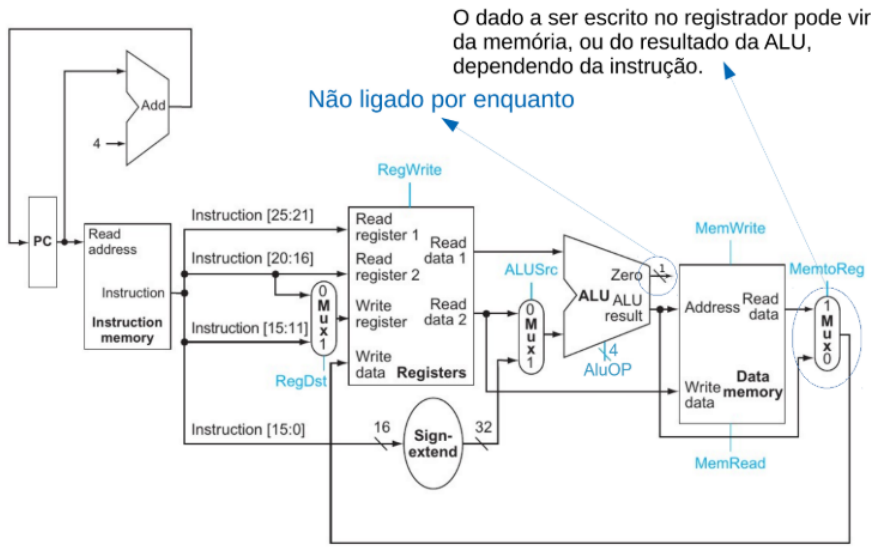
MULTIPLEXADORES EXTRAS



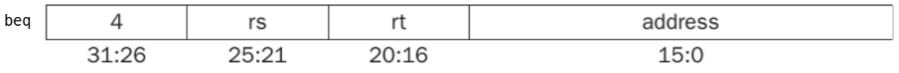
Caminho de dados



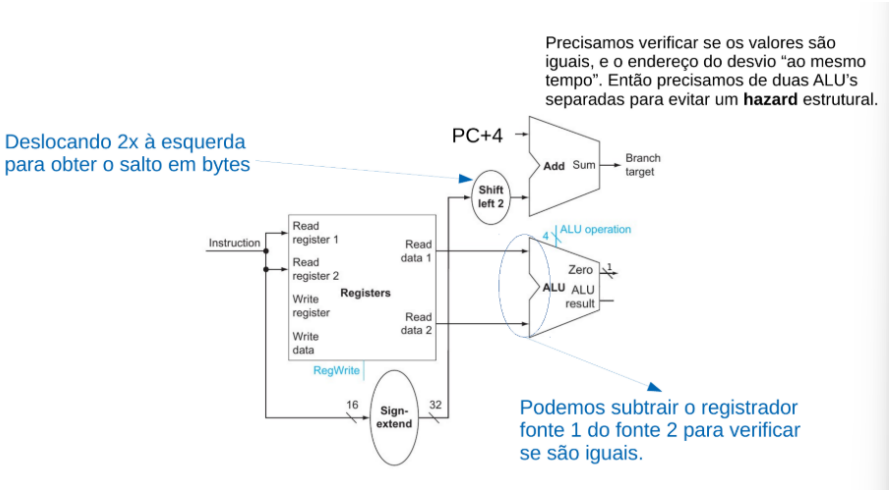




Desvio beq (Branch Equal)

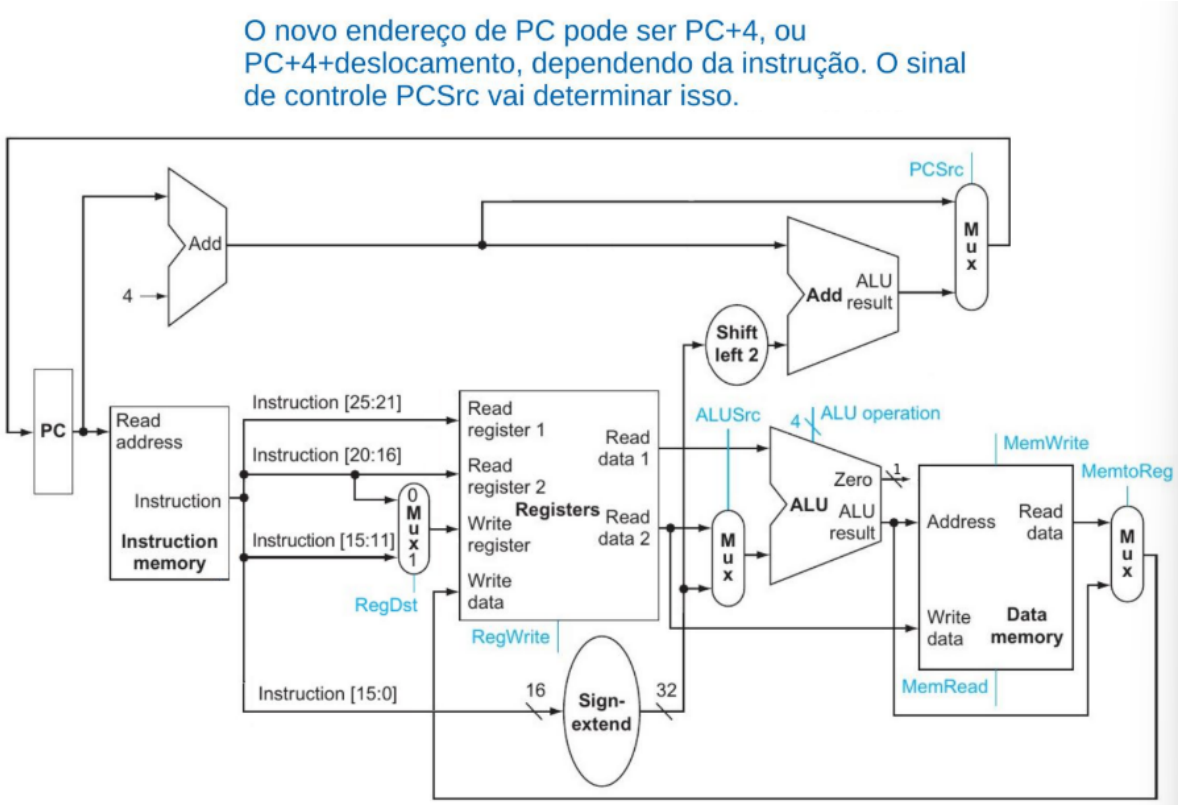


- Um desvio `beq` soma o valor de deslocamento (ques está no campo do imediato) ao PC+4 caso os registradores rs e rt sejam iguais
- Devemos deslocar 2x à esquerda para multiplicar por 4, para então obtermos o deslocamento em bytes que será armazenado em PC



- Para incluir o caminho do beq é preciso:
  - Deve haver um multiplexador para escolher entre PC+4 ou PC+4+deslocamento
    - Este multiplexador é controlador pelo sinal PCSrc
    - O PCSrc é definido pelo controlador com base na sa´ıda Zero da ALU

Caminho de dados



Controle da ALU

ALU OPERATION

- É um sinal de 4 bits
- É sinal enviado para a ALU que dependerá da instrução
- `lw` e `sw` (calculam endereço) → **Adição**
- `beq` (verifica se os valores são iguais) → **Subtração**
- `Intrução tipo R` → operação definida pelo campo **funct** de 6 bits da instrução

ALU control lines	Function
0000	AND
0001	OR
0010	add
0110	subtract
0111	set on less than
1100	NOR

SINAIS DE ENTRADA NA UNIDADE DE CONTROLE

- A unidade de controle da ALU vai receber como entrada um sinal de 2 bits (ALUOp) que define o tipo de instrução e recebe
- Recebe também como entrada o sinal do campo funct

ALUOp

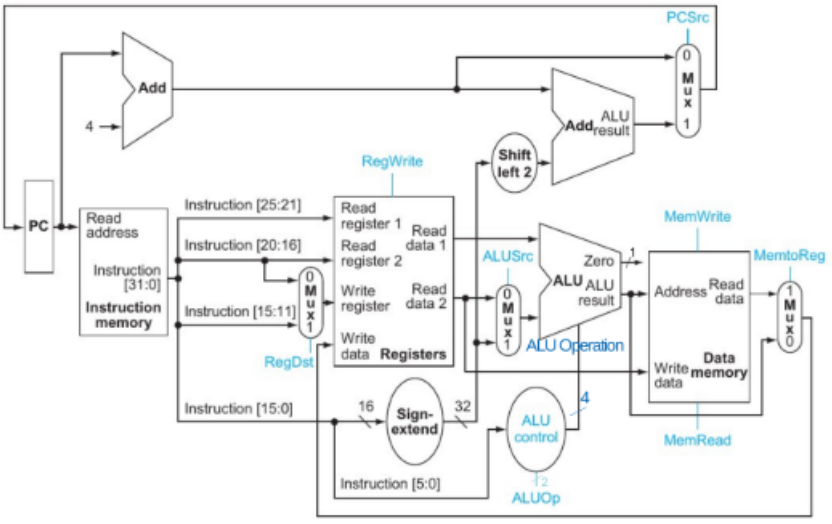
- 00 → indica que a operaçãoo é uma adição (para lw e sw)
- 01 → indica que a operação é uma subtração (para beq/bne) g
- 10 → indica que a operação vai ser definida pelo campo funct (tipo-R)

ALUOp e Funct Field: entradas do controle da ALU

ALU control input (ALU Operation): saída para a ALU

- Os dois primeiros campos do funct não são relevantes na escolha da operação
  - Mas, todos os campos de funct são enviados para o controle da ALU, pois uma implementação mais ampla do MIPS pode precisar desses campos

MONTAGEM

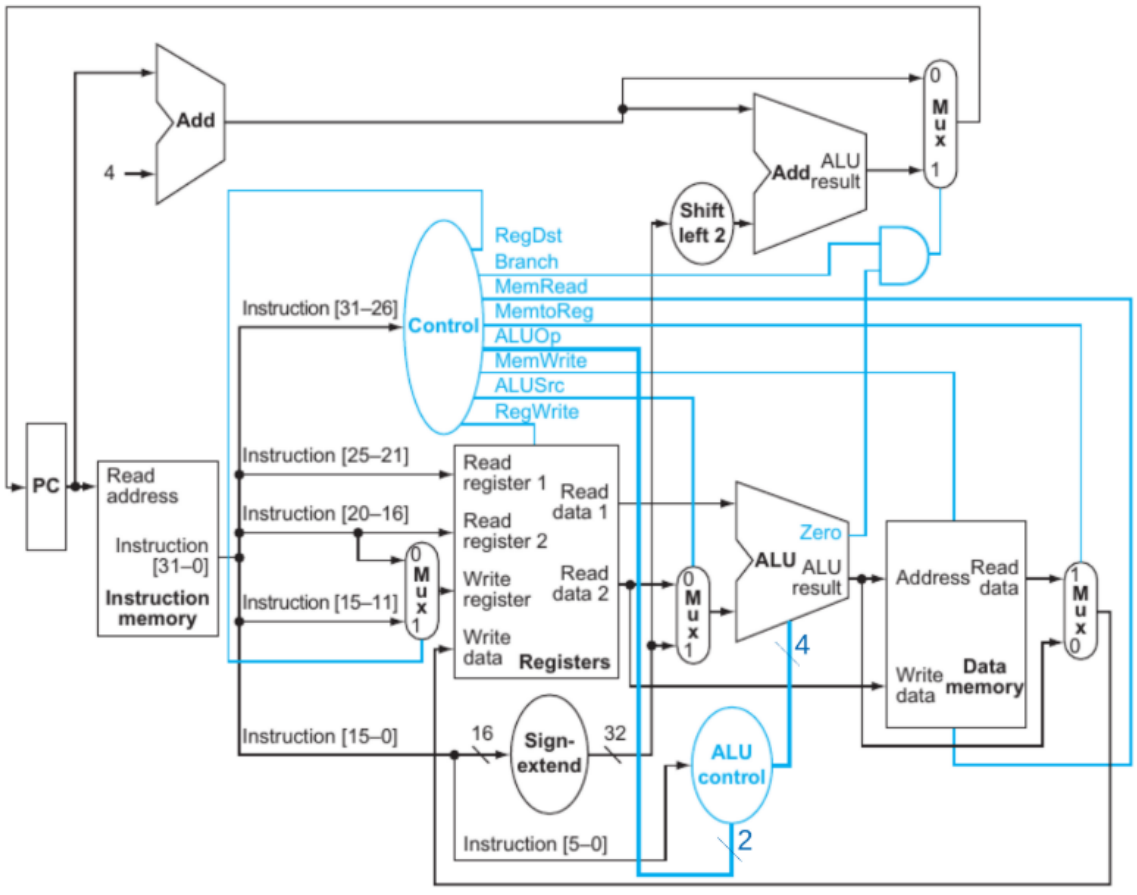


- O controle da ALU gera o sinal **ALU Operation** de 4 bits
- Mas depende de um sinal **ALUOp** de 2 bits
- O sinal ALUOP, será gerado pela unidade de controle principal

Sinal	Efeito esperado quanto 0	Efeito esperado quanto 1
RegDst	O núm. do reg. de destino deve vir do campo rt (bits [20:16]).	O núm. do reg. de destino deve vir do campo rd (bits [15:11])
RegWrite	Nada acontece	Os dados enviados ao banco de registradores são escritos no endereço do registrador de escrita.
ALUSrc	O segundo operando da ALU vêm do segundo registrador fonte lido do banco de registradores.	O segundo operando da ALU vêm do campo imediato da instrução (16 bits estendidos para 32).
PCSrc	PC recebe PC+4	PC recebe o endereço de desvio calculado
MemRead	Nada acontece	O conteúdo da memória no endereço especificado é enviado para a saída de leitura da memória
MemWrite	Nada acontece	O conteúdo da memória no endereço especificado é substituído pelo dado na entrada “WriteData” da memória
MemtoReg	O valor a ser escrito no registrador de destino deve vir da ALU	O valor a ser escrito no registrador de destino deve vir da Memória

Unidade de controle principal

- Vai cuidas das 7 linhas de controle (de 1 bit) e do sinal de controle de dois bits do ALUOp



- Nossa unidade de controle é simples e pode ser to dipo “hardwired”

- Definida com portas lógicas e de comportamento fixo