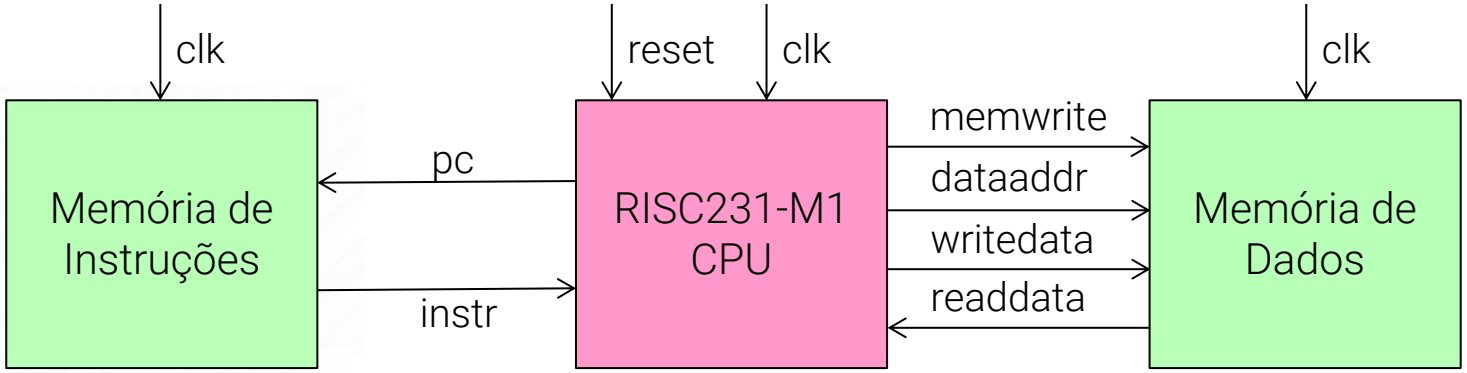
GCET231 Circuitos Digitais II

Projeto do Processador RISC231-M1

João Carlos Nunes Bittencourt

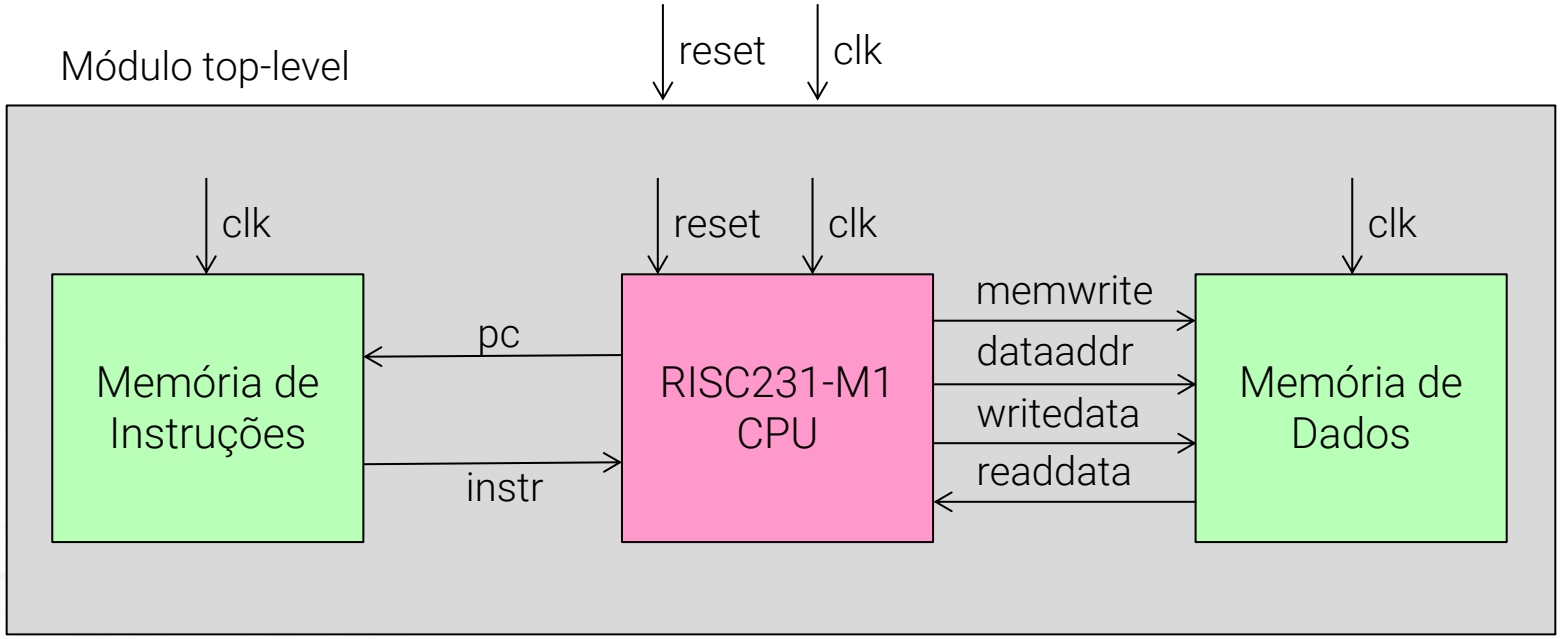
Universidade Federal do Recôncavo da Bahia

Visão Geral


clk
reset
clk
clk
Memória de Instruções
pc
instr
RISC231-M1 CPU
memwrite
dataaddr
writedata
readdata
Memória de Dados

Visão Hierárquica

Top-level

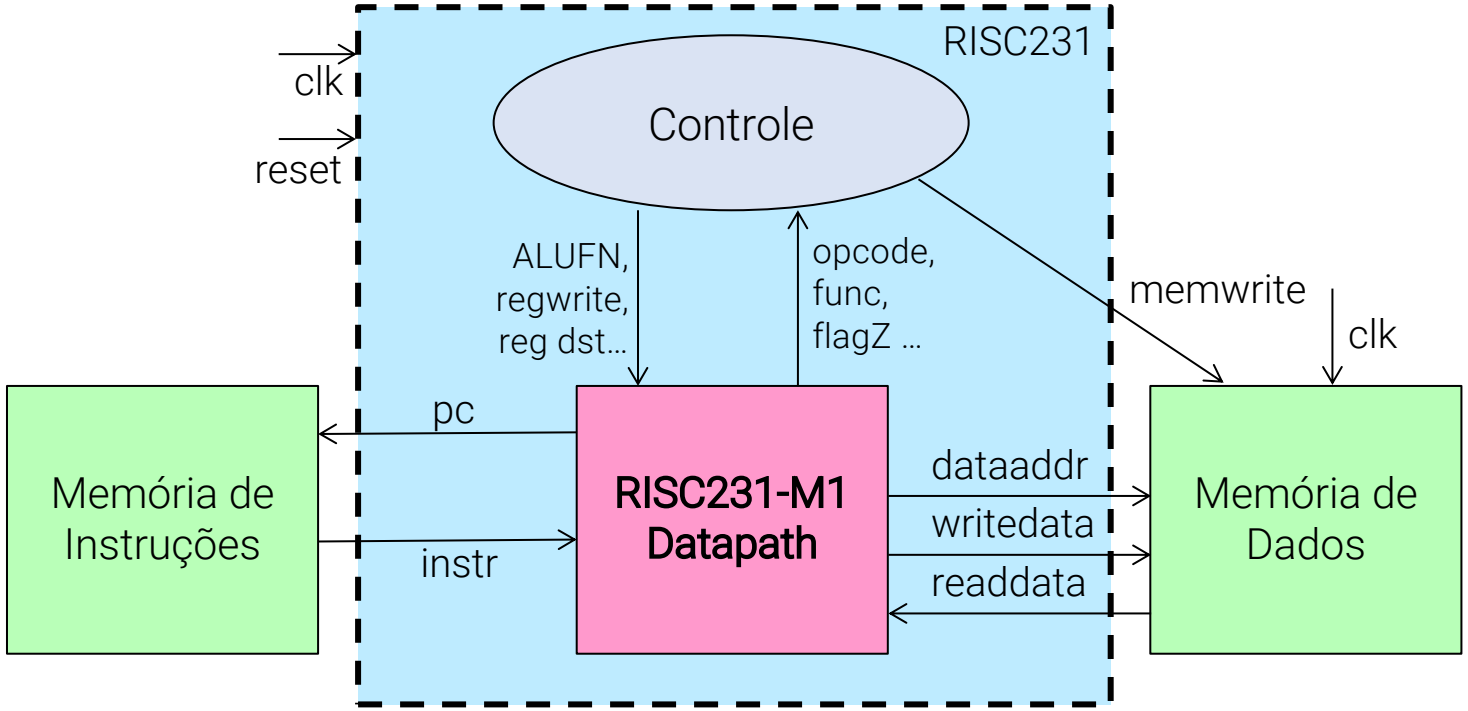


Um Nível Abaixo

Por dentro do RISC231-M1

Datapath – Caminho de Dados

Componentes que armazenam ou processam os dados dentro do processador. Composto de registrados, ALU, multiplexadores, extensor de sinal, etc.



Um Nível Abaixo

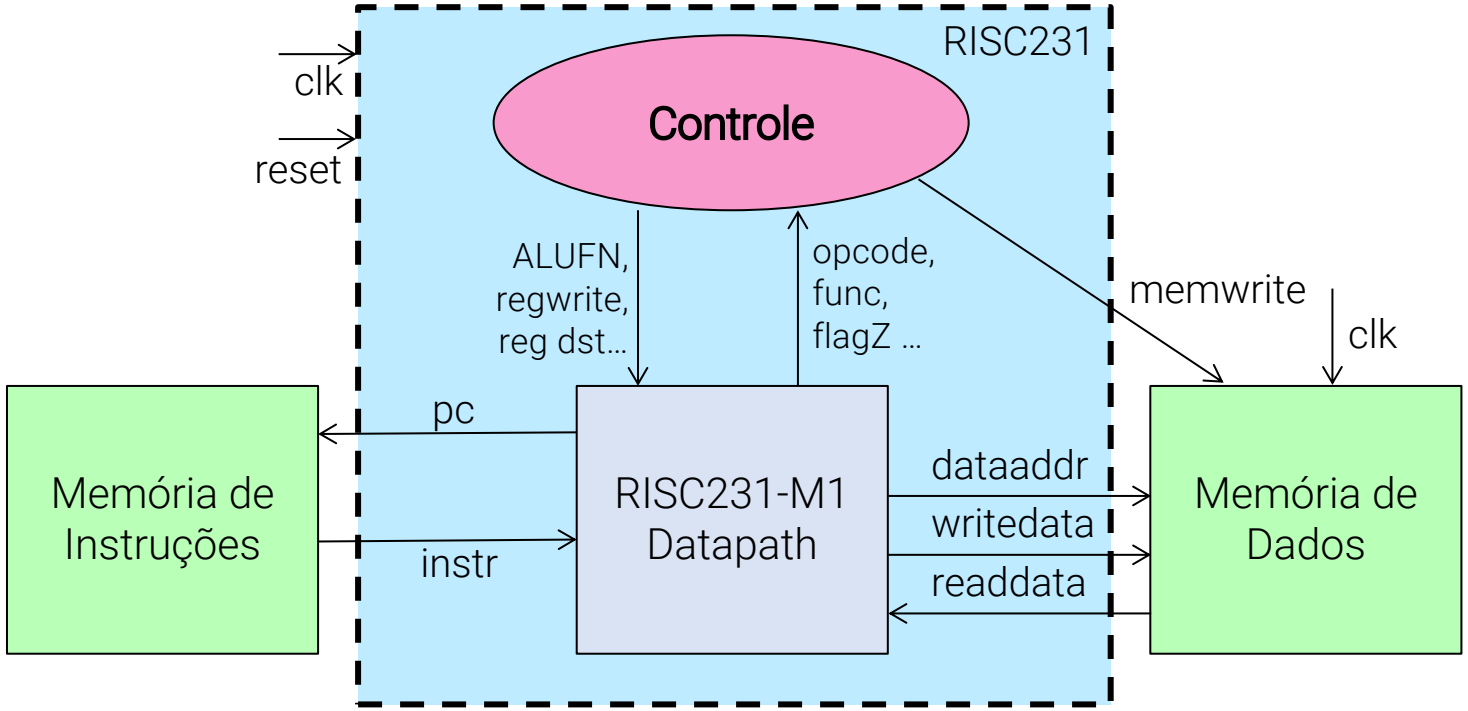
Por dentro do RISC231-M1

Datapath – Caminho de Dados

Componentes que armazenam ou processam os dados dentro do processador. Composto de registrados, ALU, multiplexadores, extensor de sinal, etc.

Controle

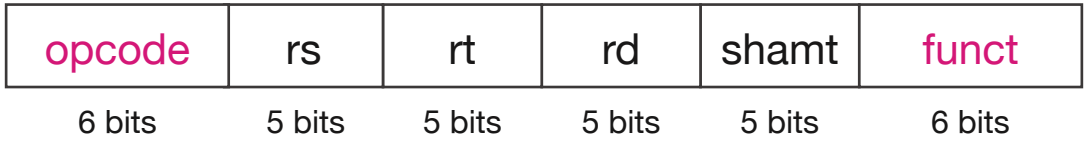
Componentes que dizem ao caminho de dados o que fazer e quando. Implementado na forma de lógica de controle, FSM ou tabelas de busca



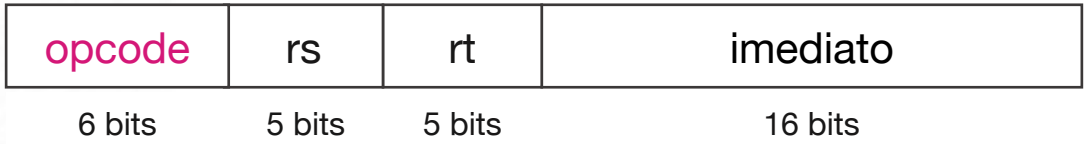
Conjunto de Instruções RISC231-M1

Formatos de Instrução

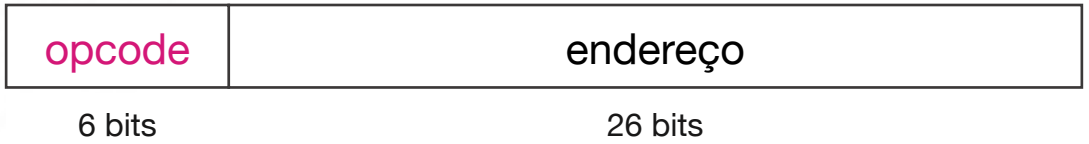
Tipo-R



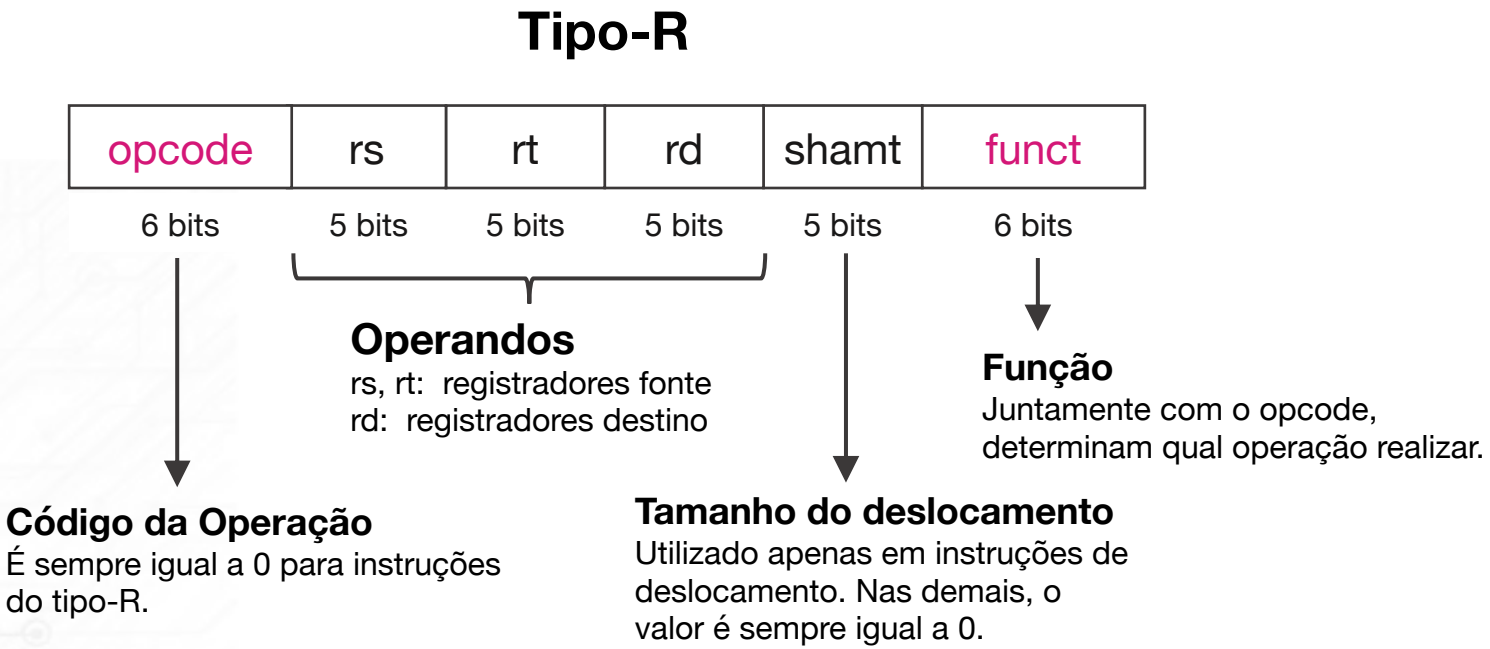
Tipo-I



Tipo-J



Instruções do Tipo-R



Instruções do Tipo-R

Exemplos

Código Assembly

```
add $s0, $s1, $s2
sub $t0, $t2, $t5
```


Observe a ordem dos registradores
`add rd, rs, rt`

Valores dos Campos

opcode	rs	rt	rd	shamt	funct
0	17	18	16	0	32
0	11	13	8	0	34
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits

Códigos de Máquina

opcode	rs	rt	rd	shamt	funct	
00000	10001	10010	10000	00000	100000	0x02328020
00000	01011	01101	01000	00000	100010	0x016D4022
6 bits	5 bits	5 bits	5 bits	5 bits	6 bits	

Instruções do Tipo-I

Registradores
Nas instruções tipo-I, o rt atua como registrador destino

Código da Operação

Valor Constante
Constante de 16 bits que vai diretamente na instrução. Representado em 2's compl.

Instruções do Tipo-I

Exemplo

Código Assembly

```
addi $s0, $s1, 5
addi $t0, $s3, -12
lw   $t2, 32($0)
sw   $s1, 4($t1)
```

Observe a ordem diferente dos registradores

```
addi rt, rs, imm
lw   rt, imm(rs)
sw   rt, imm(rs)
```

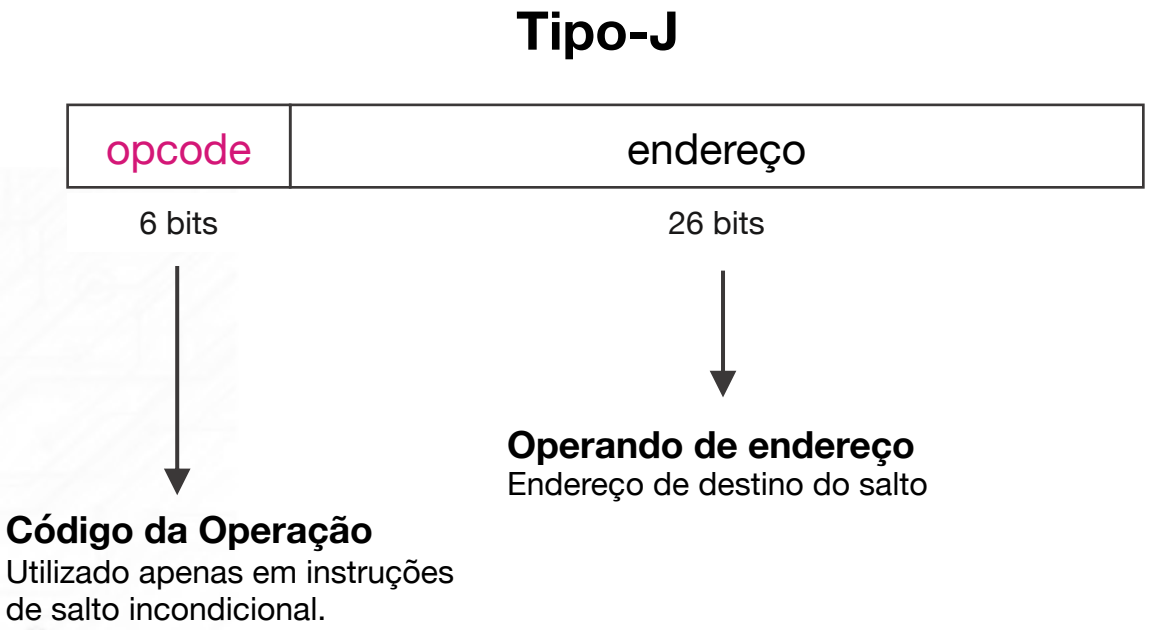
Valores dos Campos

opcode	rs	rt	imediato
8	17	16	5
8	19	8	-12
35	0	10	32
43	9	17	4
6 bits	5 bits	5 bits	16 bits

Código de Máquina

opcode	rs	rt	imediato	
001000	10001	10000	0000_000_0000_0101	0x22300005
001000	10011	01000	1111_1111_1111_0100	0x2268FFF4
100011	00000	01010	0000_0000_0010_0000	0x8C0A0020
101011	01001	10001	0000_0000_0000_0100	0xAD310004
6 bits	5 bits	5 bits	16 bits	

Instruções do Tipo-J



Projeto Bottom-Up do RISC231-M1

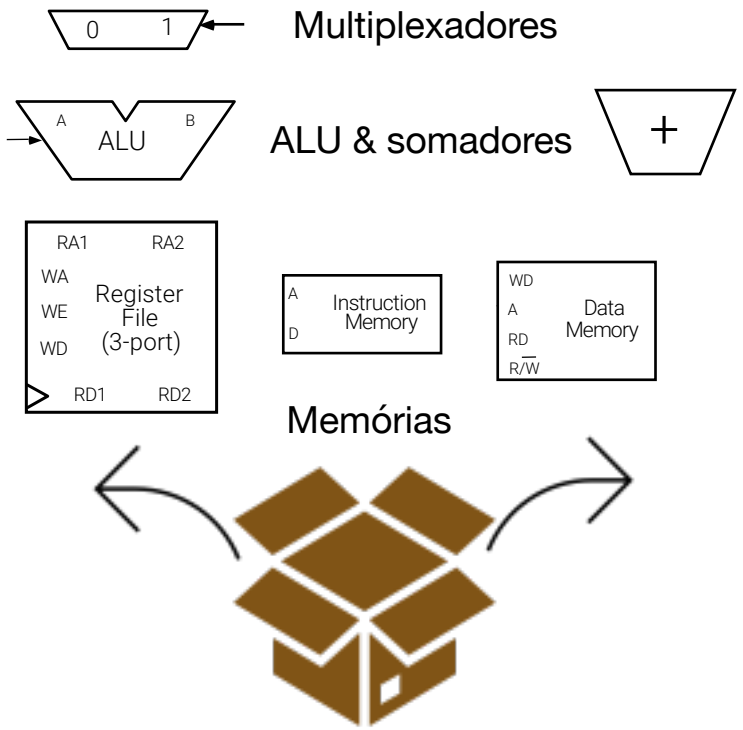
Abordagem de Projeto

Iterativo e Incremental

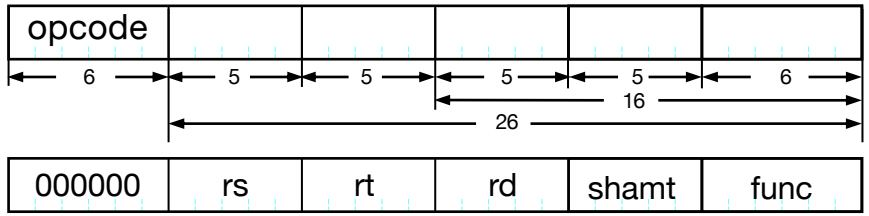
Nós iremos implementar circuitos para cada tipo de instrução, e depois juntá-las (usando multiplexadores, resigstradores, etc.)

Etapas

1. Instruções ALU de 3 operandos
2. Instruções ALU com imediato
3. Leitura e Gravação na Memória de Dados
4. Saltos e Tomadas de Decisão
5. Comparação
6. Carregamento de Imediato



Arquitetura de Instruções RISC231-M1



Tipo-R: ALU com operandos registradores

`Reg[rd] ← Reg[rs]` **op** `Reg[rt]`

Tipo-I: ALU operando com constante

`Reg[rt] ← Reg[rs]` **op** `SEXT(imediato)`

Tipo-I: Load e Store

```
Reg[rt] ← Mem[Reg[rs] + SEXT(imediato)]
Mem[Reg[rs] + SEXT(imediato)] ← Reg[rt]
```

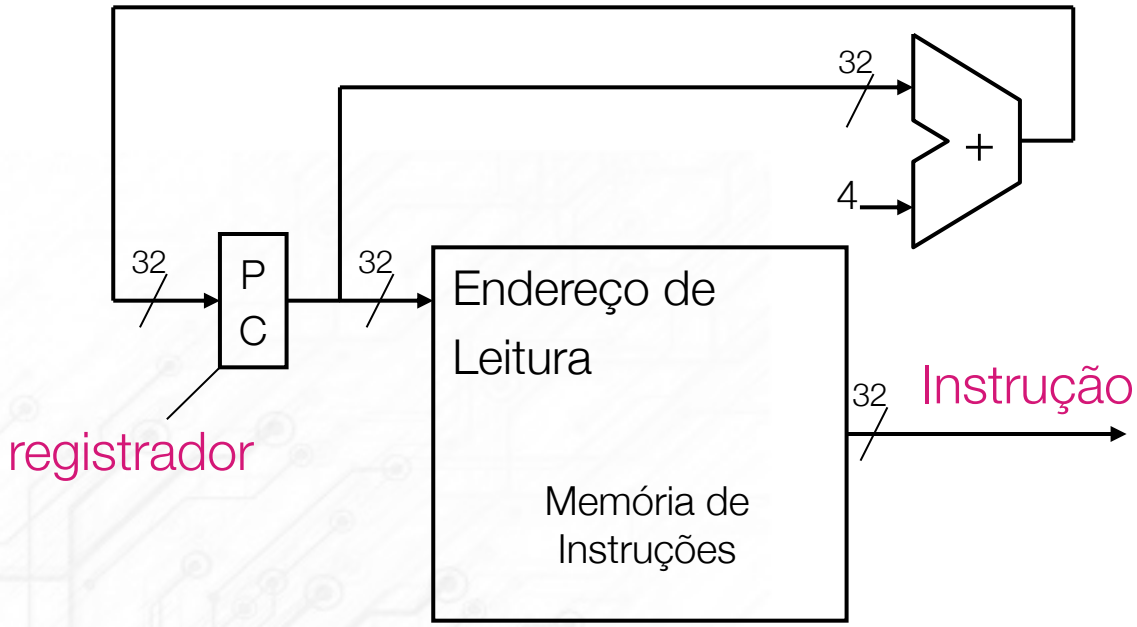
Tipo-I: Instruções de Branch

```
if (Reg[rs] == Reg[rt]) PC ←  PC + 4 + 4*SEXT(imediato)
if (Reg[rs] != Reg[rt]) PC ←  PC + 4 + 4*SEXT(imediato)
```

Tipo-J: Instruções de Salto

`PC ← (PC & 0xf0000000) | 4*(imediato)`

Buscando Instruções de Forma Sequencial



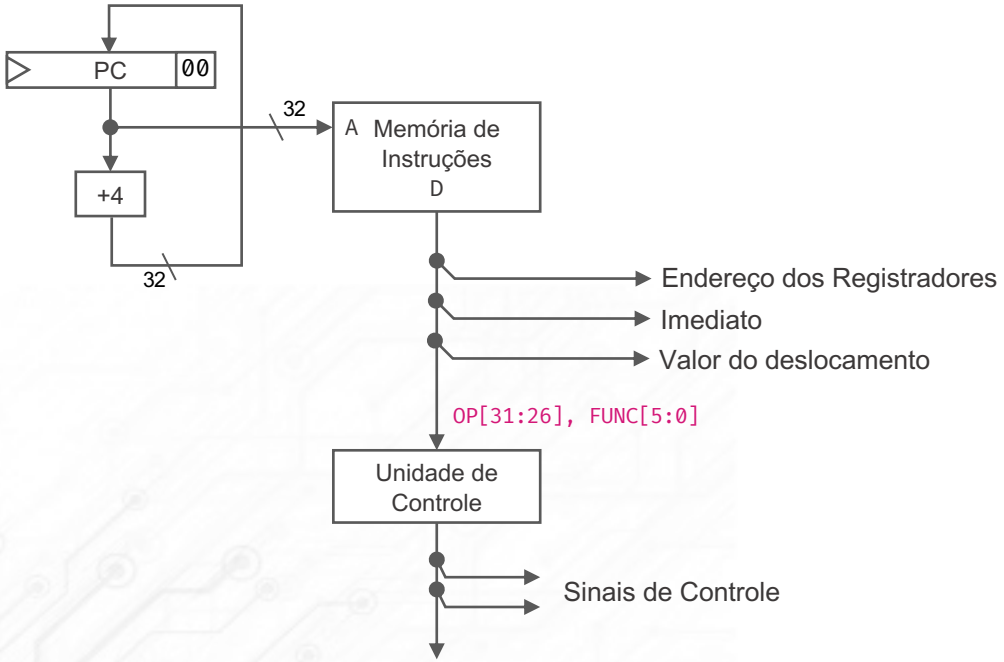
Contador de Programas PC

Utilize o PC como endereço da memória de instruções. Soma-se 1 bit a PC, carregando novos valores ao final do ciclo.

Memórias no FPGA

Usaremos um deslocamento de dois bits no valor de PC para endereçar palavras, no lugar de bytes.

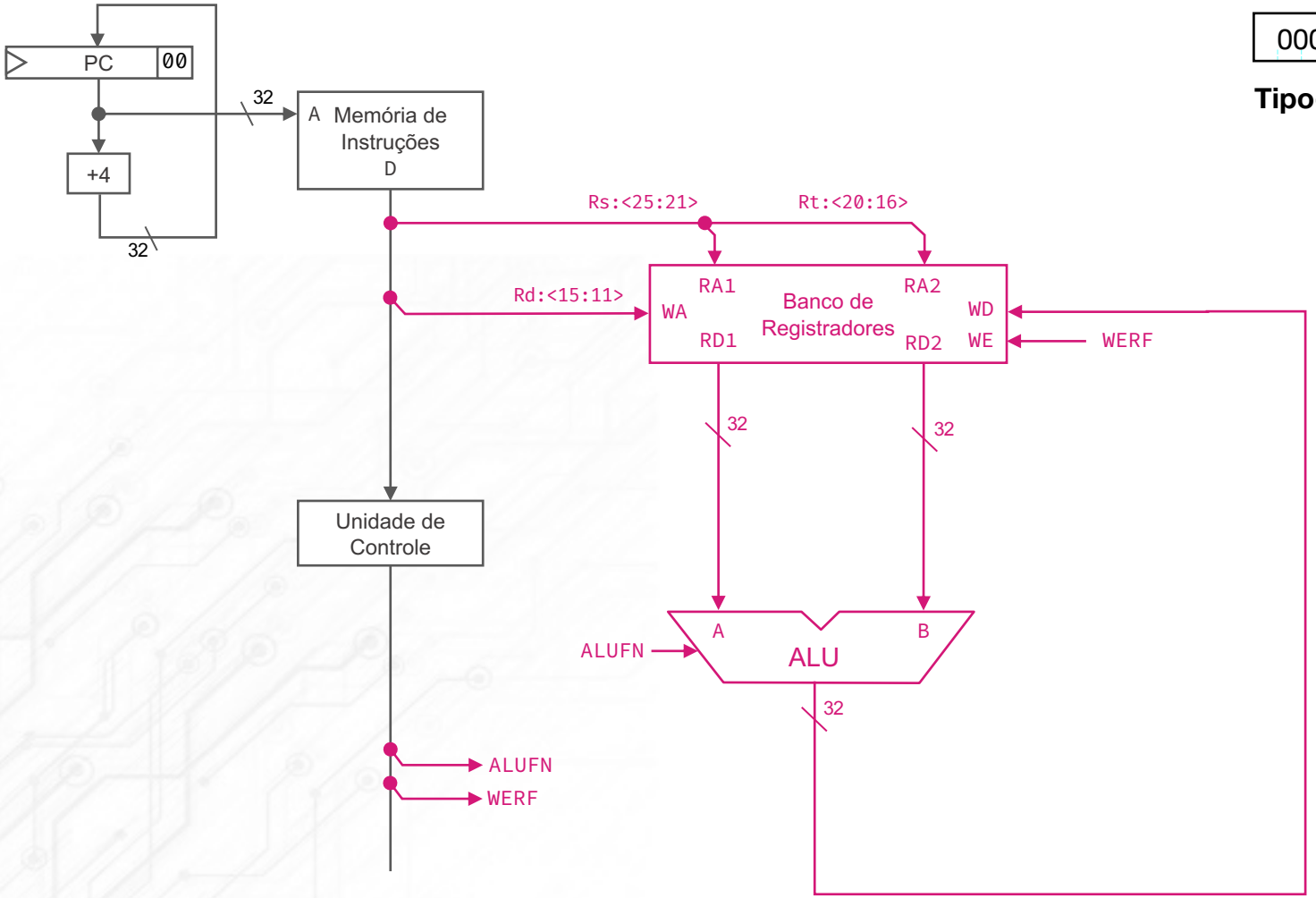
Leitura e Decodificação da Instrução



Decodificação da Instrução

Os outros campos da instrução são utilizados para decodificação. Utilize os bits <31:26> e <5:0> para produzir os sinais de controle

Operações na ALU com 3 Operandos

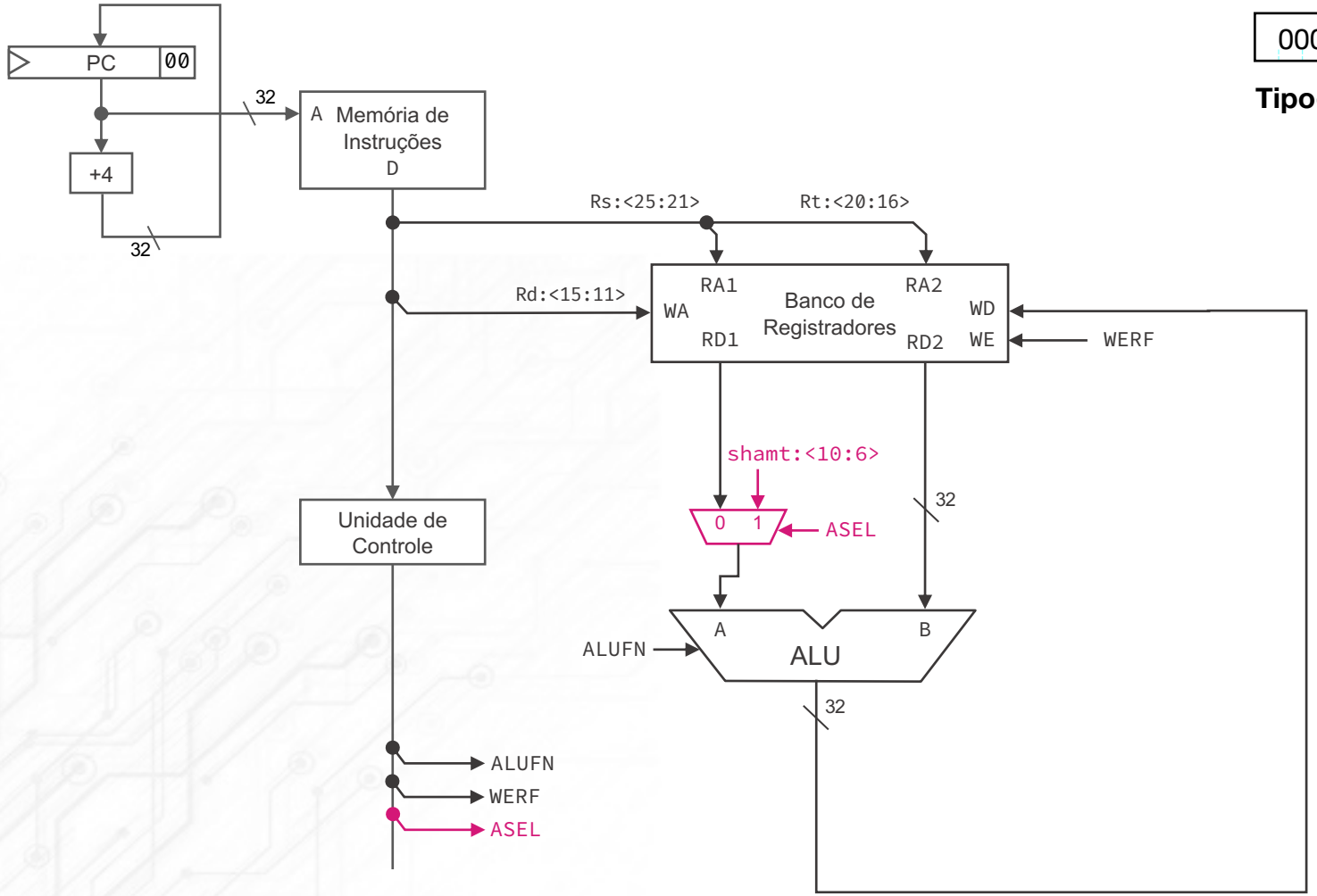


000000	rs	rt	rd	shamt	func

Tipo-R: ALU com operandos registradores

`Reg[rd] ← Reg[rs] op Reg[rt]`

Operações de Deslocamento



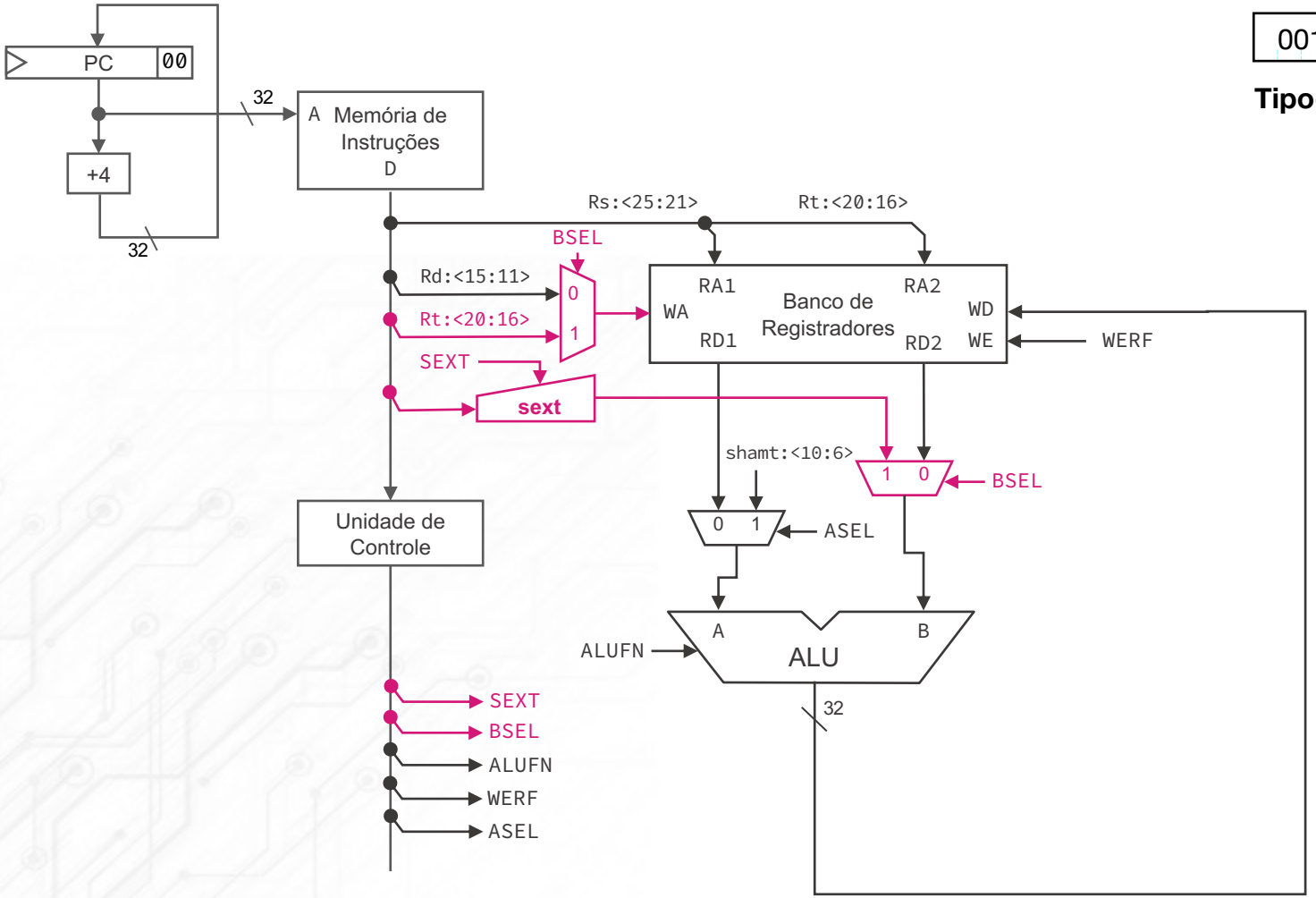
000000	rs	rt	rd	shamt	func

Tipo-R: ALU com operandos registradores

`SLL: Reg[rd] ← Reg[rt] (shift) shamt`

`SLLV: Reg[rd] ← Reg[rt] (shift) Reg[rs]`

Operações na ALU com Imediato



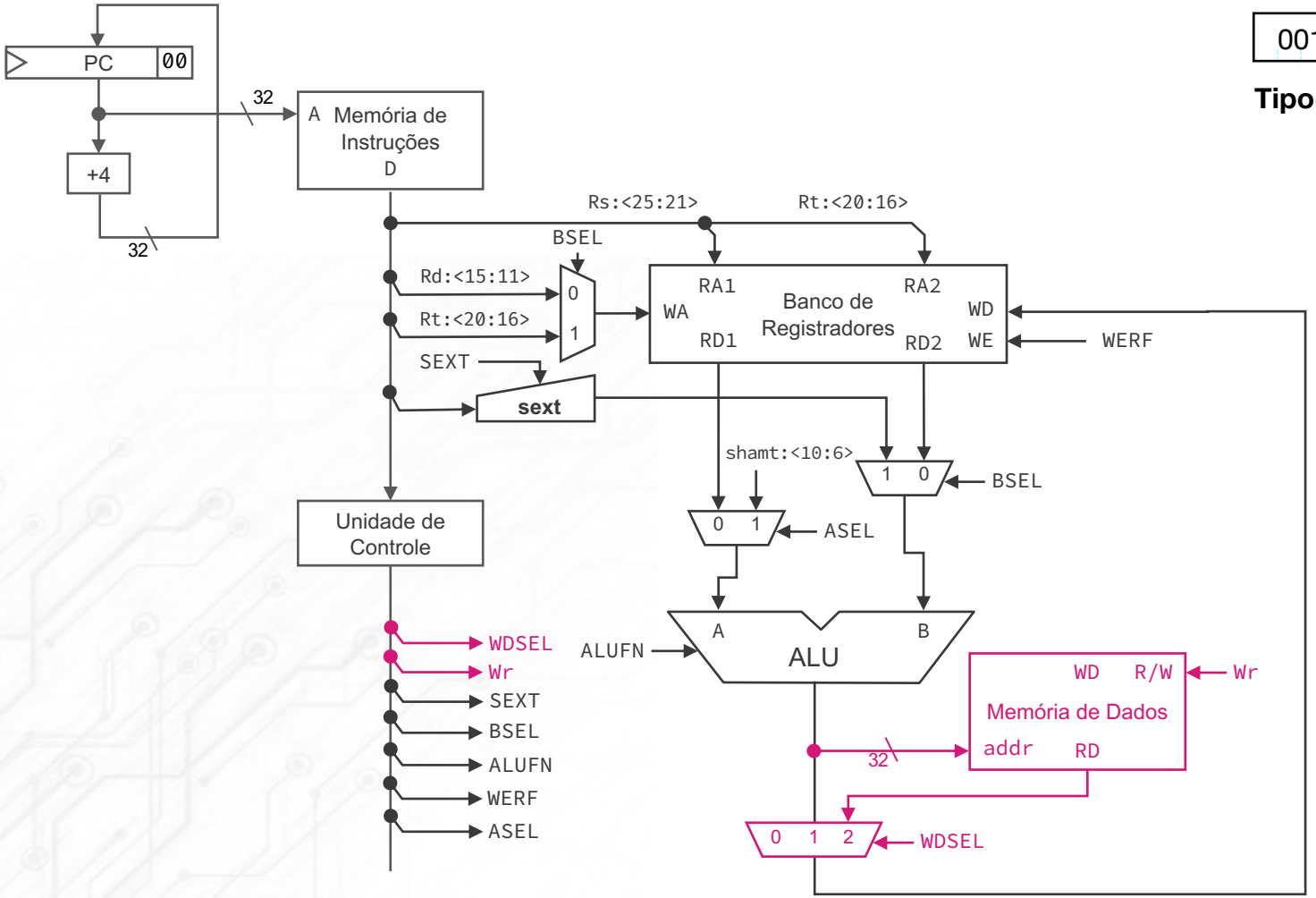
001XXX	rs	rt	imediato

Tipo-I: ALU operando com constante
`Reg[rt] ← Reg[rs]` **op** `SEXT(imediato)`

Como projetar seu sext?

Se **SEXT** for igual a 1, completa com o sinal, caso contrário, completa com zeros.

Operação de Leitura da Memória

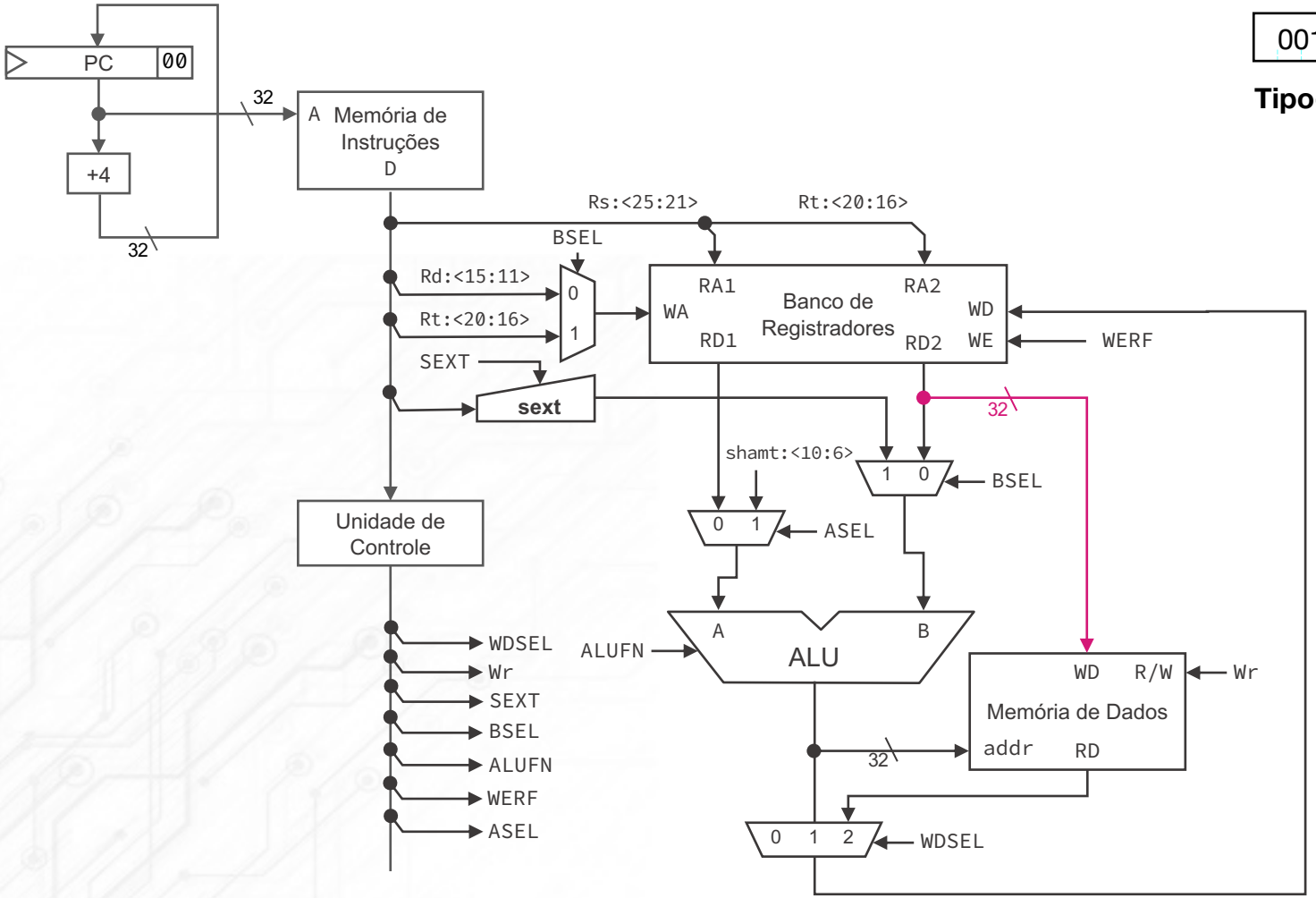


001XXX	rs	rt	imediato

Tipo-I: Load

`Reg[rt] ← Mem[Reg[rs] + SEXT(imediato)]`

Operação de Gravação na Memória



001XXX	rs	rt	imediato

Tipo-I: Store

`Mem[Reg[rs] + SEXT(imediato)] ← Reg[rt]`

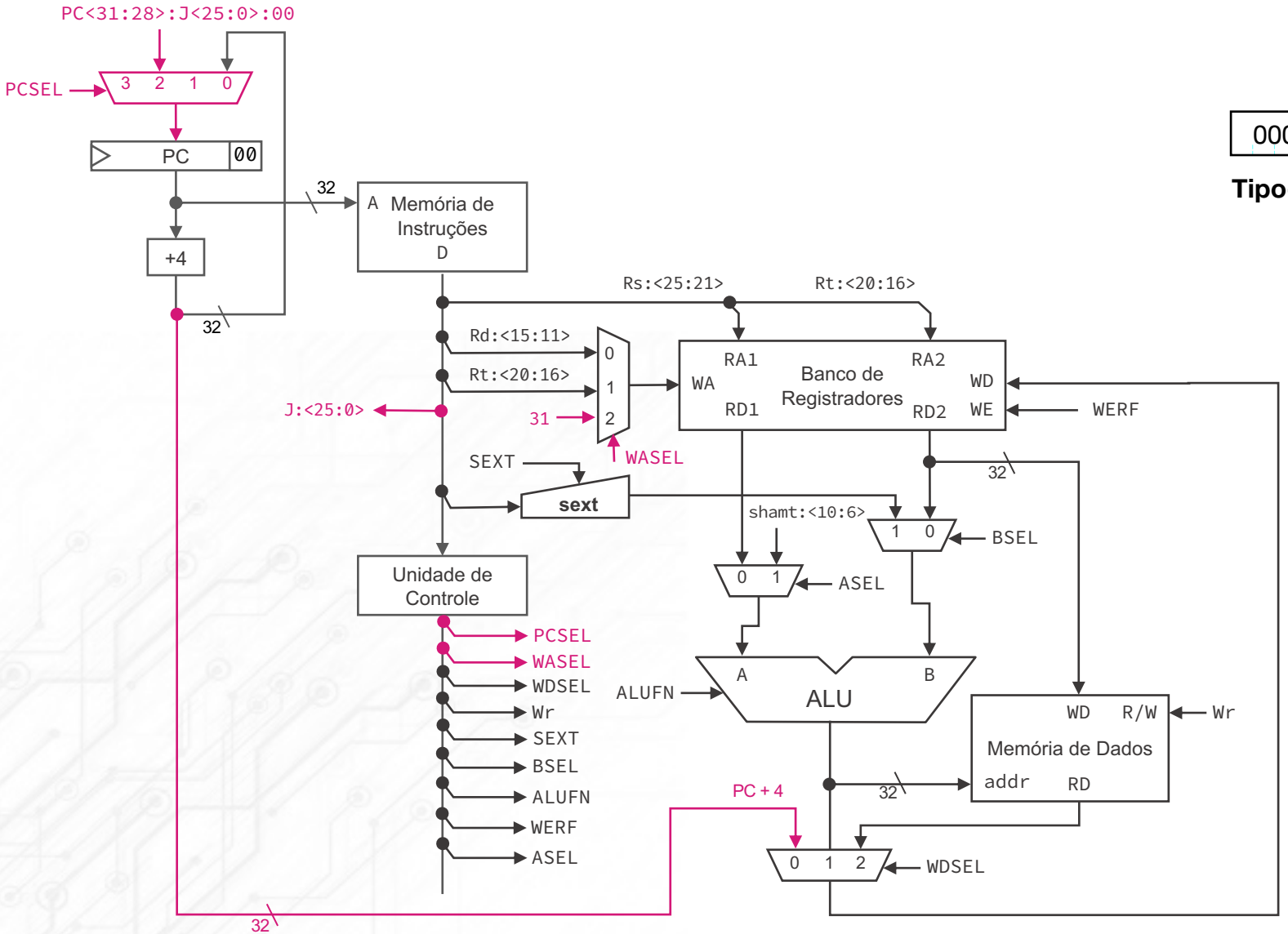
Escrita na Memória

A unidade de controle precisa ativar o sinal Wr para habilitar a escrita na memória.

WERF

Nesse momento, é importante desativar a escrita no banco de registradores.

Instruções de Salto Incondicional

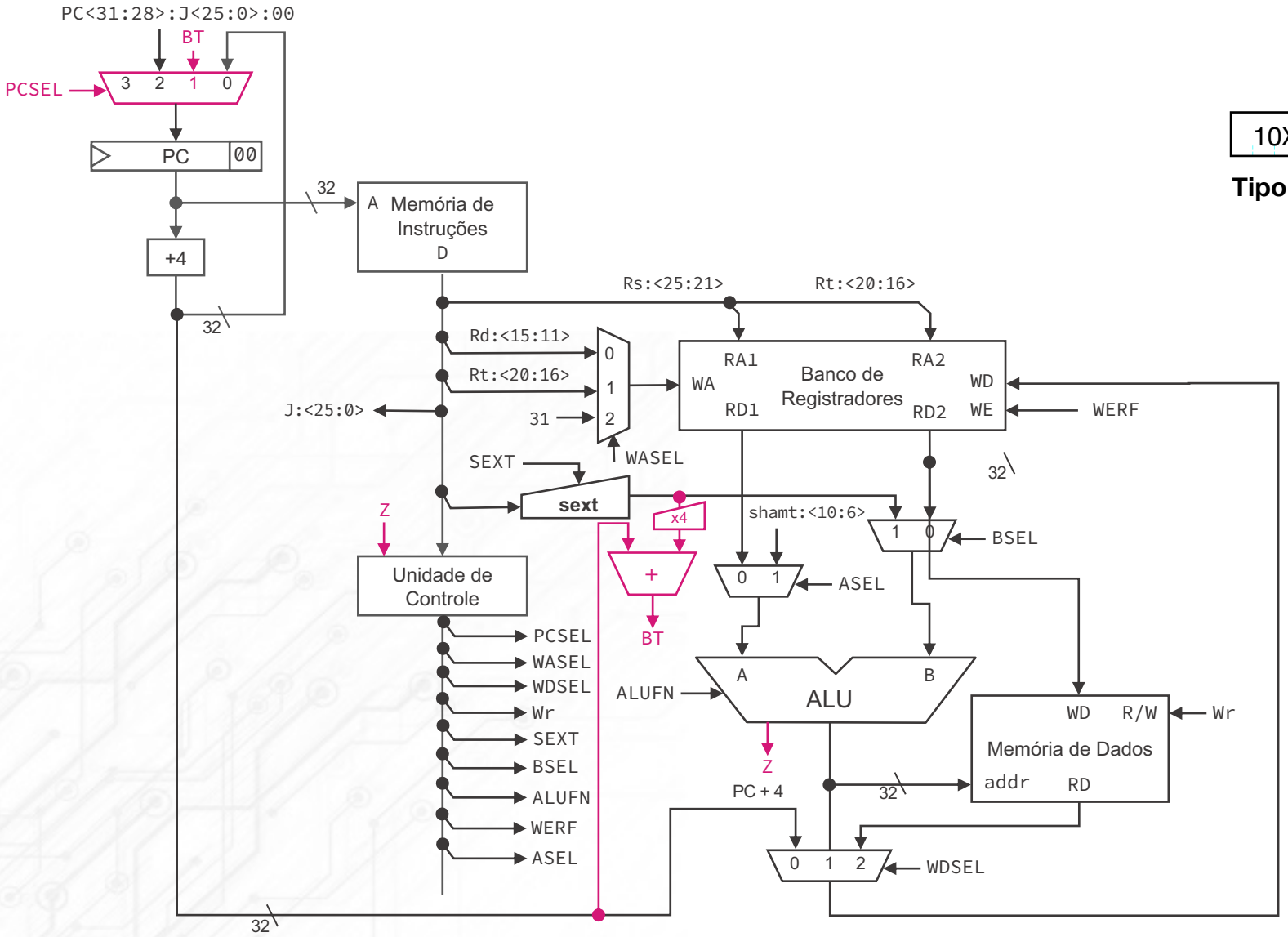


00001X	constante de 26-bit

Tipo-J: Instruções de Salto

```
J:   PC ← (PC & 0xf0000000) | 4*(imediato)
JAL: PC ← (PC & 0xf0000000) | 4*(imediato),
     Reg[31] ← PC + 1
```

Instruções de Tomada de Decisão

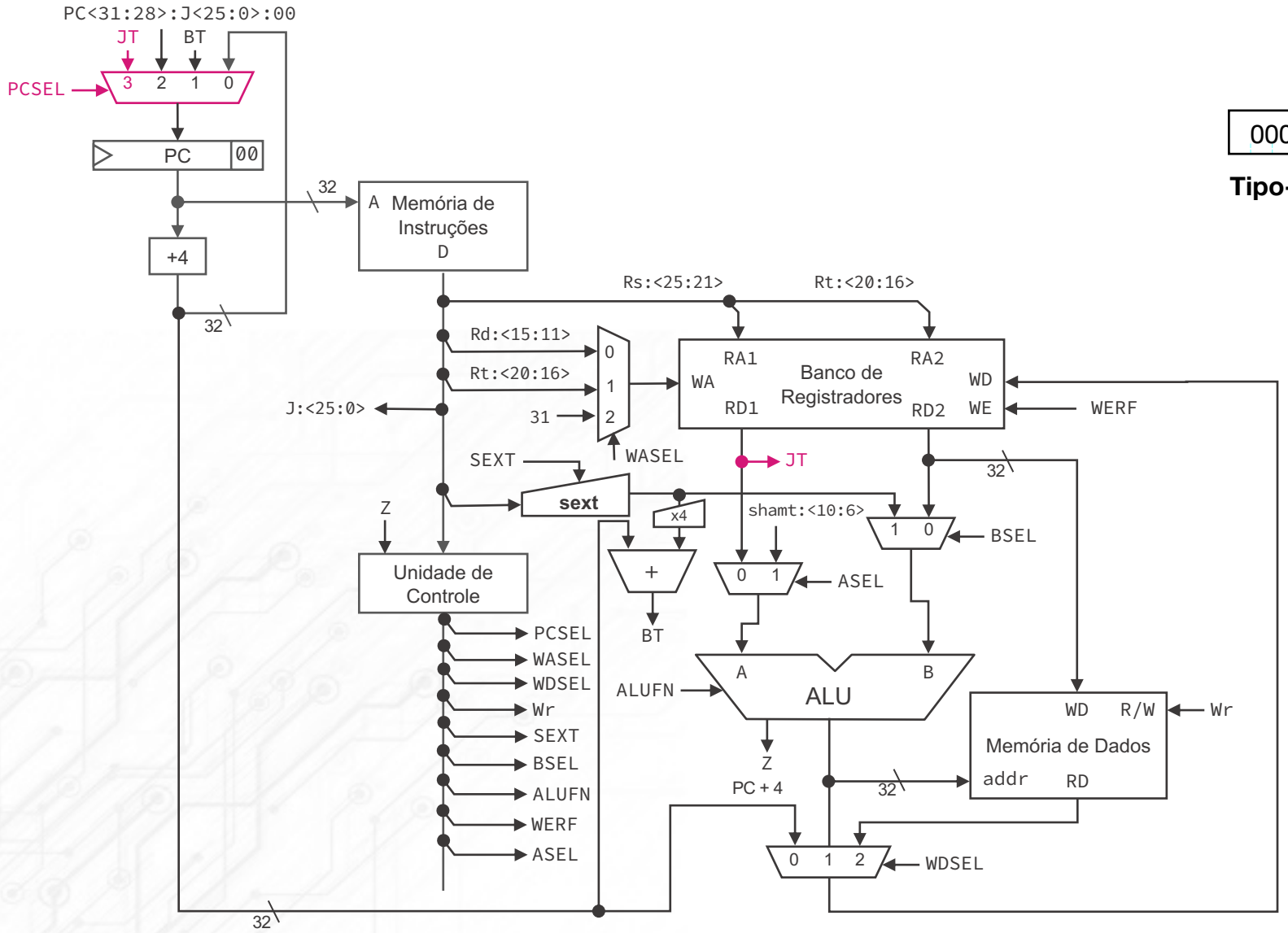


10X011	rs	rt	imediato

Tipo-I: Instruções de Branch

```
if(Reg[rs] == Reg[rt])
   PC ← PC + 4 + 4*SEXT(imediato)
if(Reg[rs] != Reg[rt])
   PC ← PC + 4 + 4*SEXT(imediato)
```

Instruções de Salto Incondicional Indireto



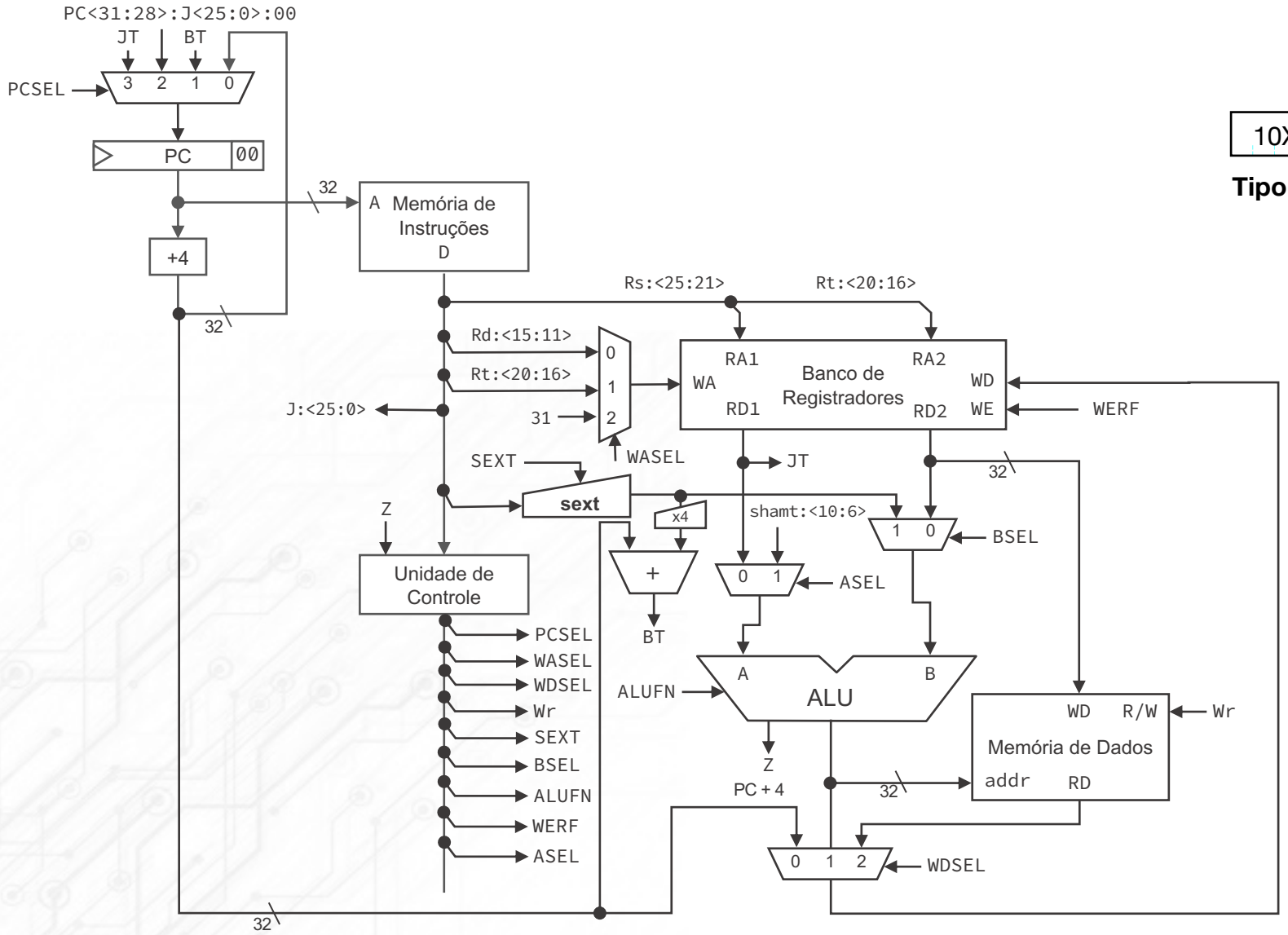
000000	rs	rt	rd	shamt	func

Tipo-R: Salto Indireto

`JR: PC ← Reg[rs]`

`JALR: PC ← Reg[rs], Reg[rd] ← PC + 1`

Instruções de Comparação



10X011	rs	rt	imediato

Tipo-I: set less then & set less then unsigned immediate

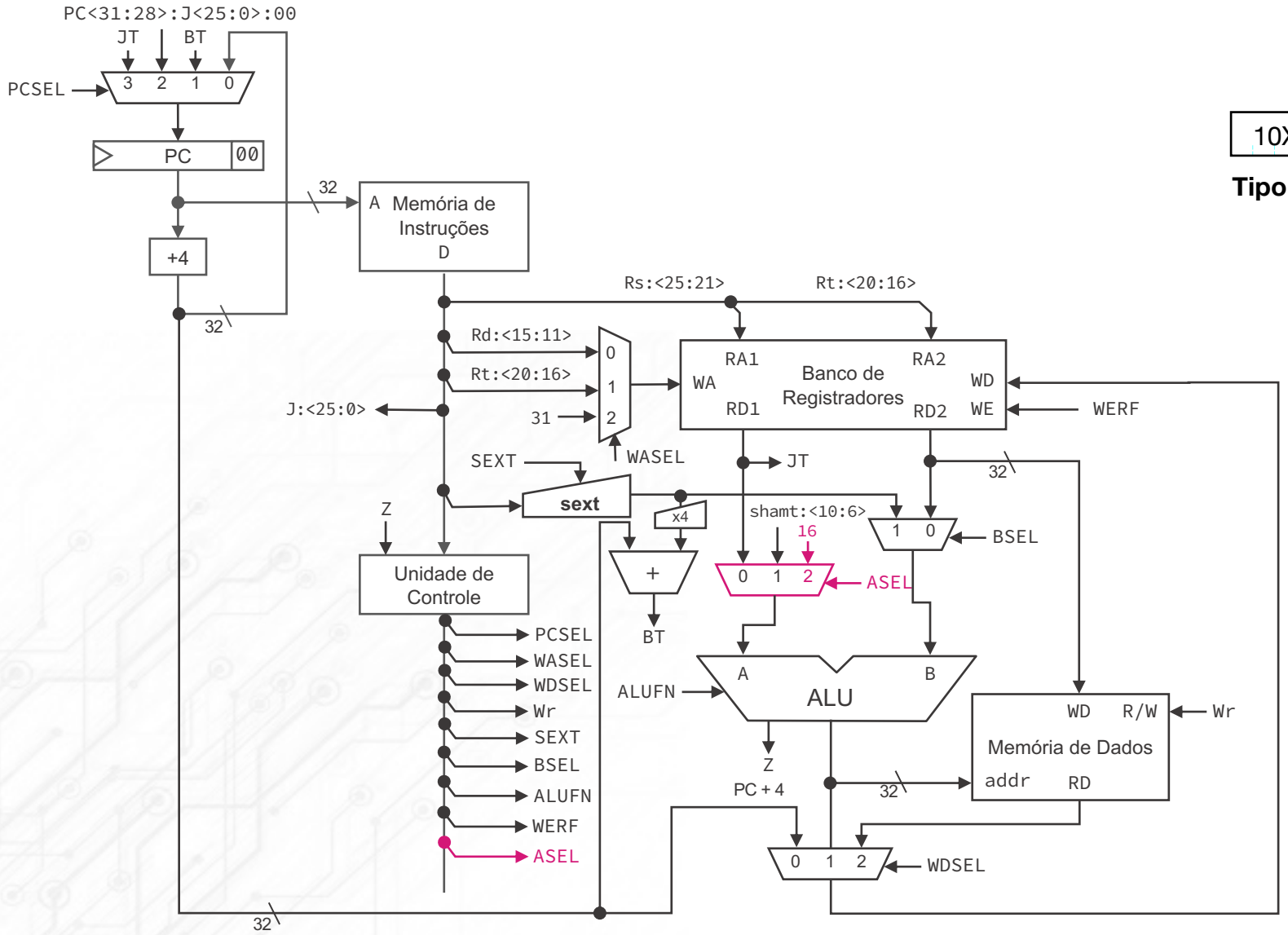
```
SLTI:  if (Reg[rs] < SEXT(imm))
         Reg[rt] ← 1;
       else Reg[rt] ←  0
SLTIU: if (Reg[rs] < SEXT(imm))
         Reg[rt] ← 1;
       else Reg[rt] ←  0
```

000000	rs	rt	rd	shamt	func

Tipo-R: set less then & set less then unsigned

```
SLT:  if (Reg[rs] < Reg[rt])
        Reg[rd] ← 1;
      else Reg[rd] ←  0
SLTU: if (Reg[rs] < Reg[rt])
        Reg[rd] ← 1;
      else Reg[rd] ←  0
```

Instruções para Carregar MSB Imediato

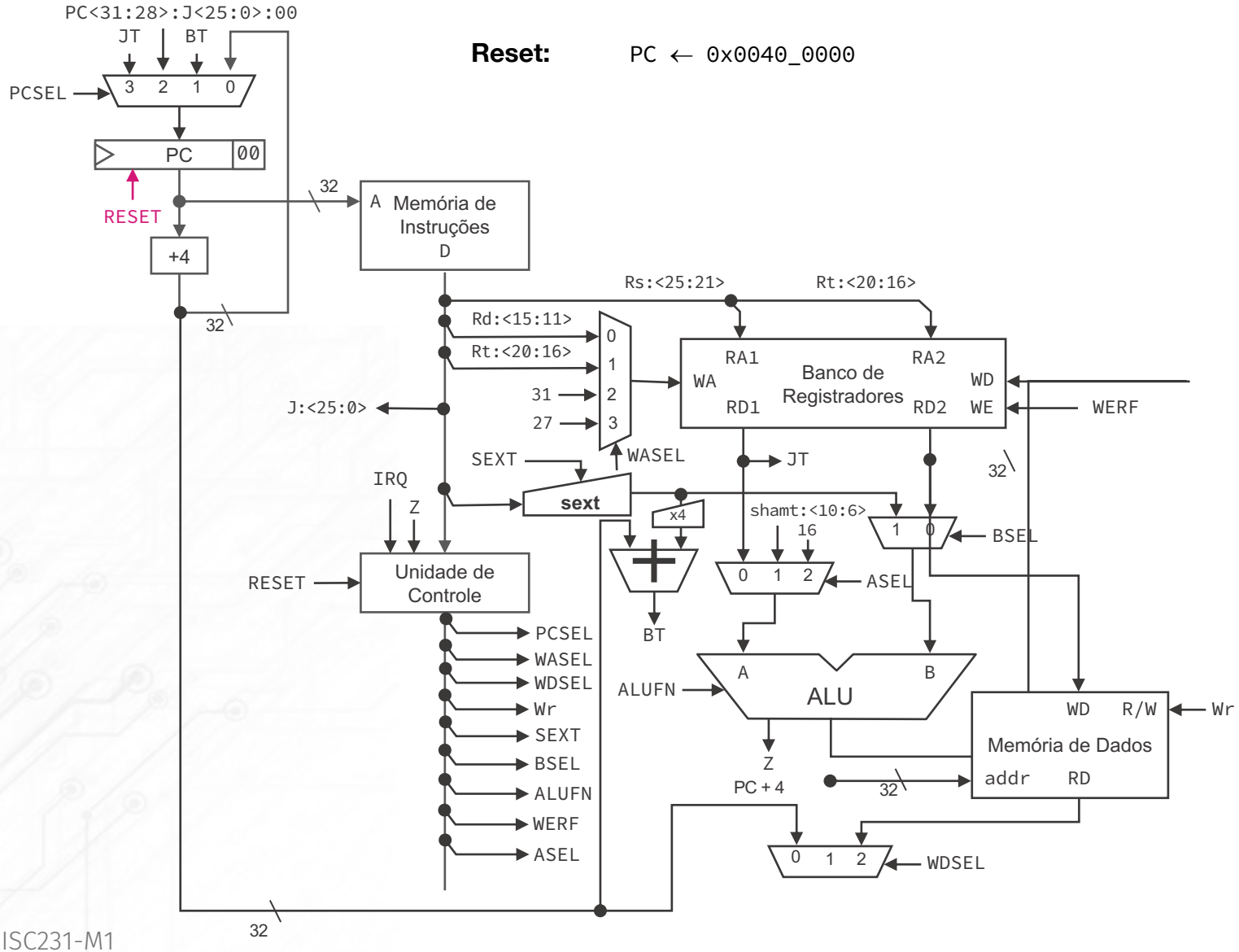


10X011	rs	rt	imediato

Tipo-I: Load upper immediate

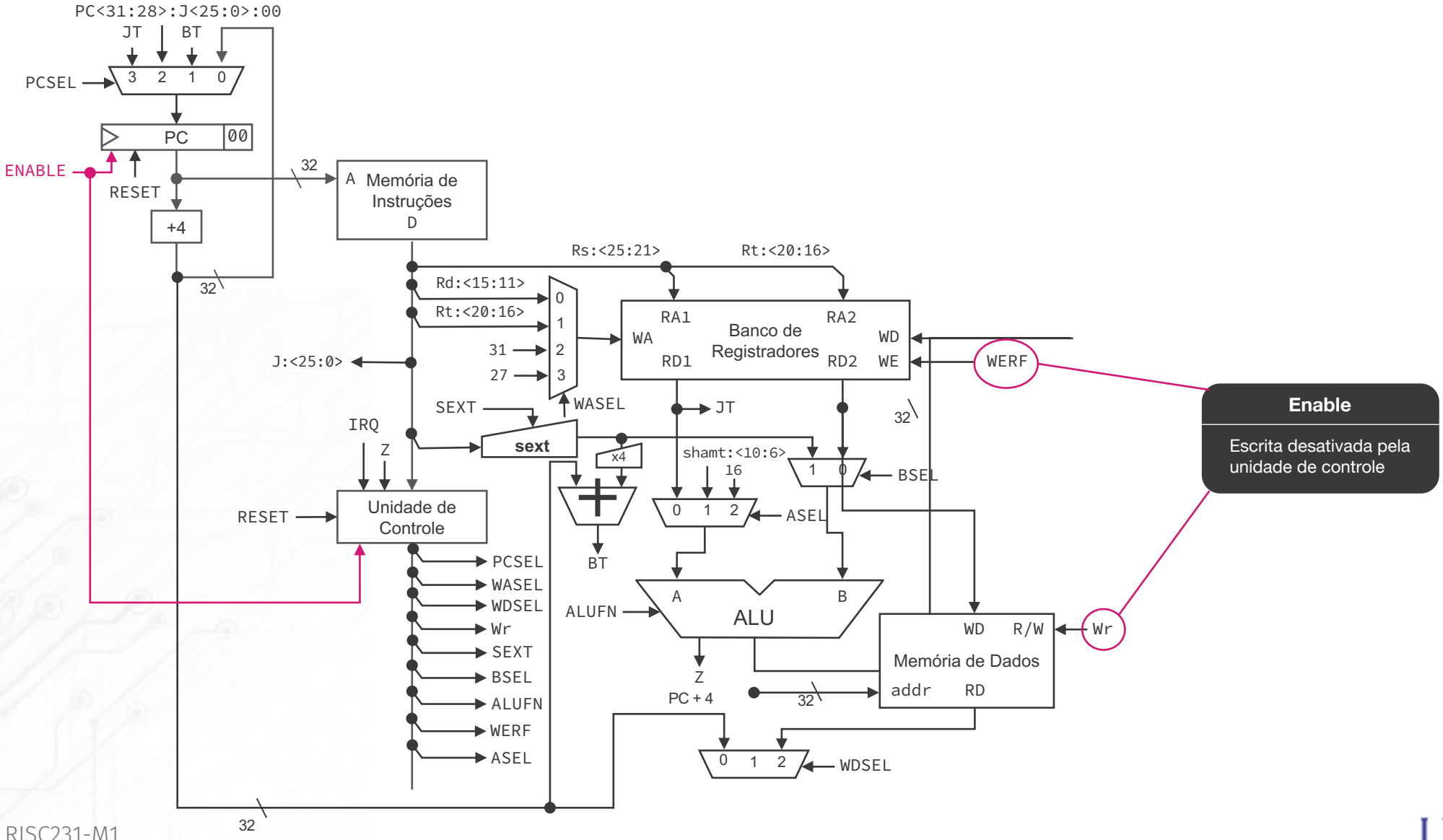
`LUI: Reg[rt] ← (imediato << 16)`

Suporte ao Reset



Reset: PC ← 0x0040_0000

Suporte ao Enable



Hierarquia do RISC231-M1