

INTEGRANTES:

EDUARDO HENRIQUE FREIRE MACHADO(2020001617) KELVIN ARAÚJO FERREIRA (2019037653)

DESCRIÇÃO DO PROCESSADOR

- Processador RISC;
- Baseado no MIPS;
- Processador de 8 bits.

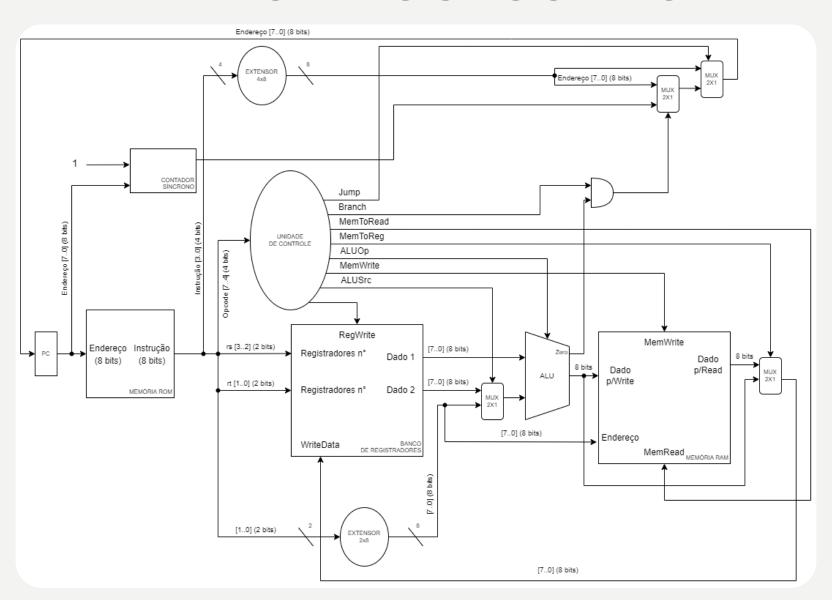
FORMATOS DAS INSTRUÇÕES

INSTRUÇÃO TIPO R									
OPCODE	RS	RT							
4 BITS	2 BITS	2 BITS							
7-4	3-2	1-0							
INSTRUÇÃO TIPO I									
OPCODE	RS	VALOR							
4 BITS	2 BITS	2 BITS							
7-4	3-2	1-0							
INSTRUÇÃO TIPO J									
OPCODE	ENDE	REÇO							
4 BITS	4 BITS								
7-4	3-0								

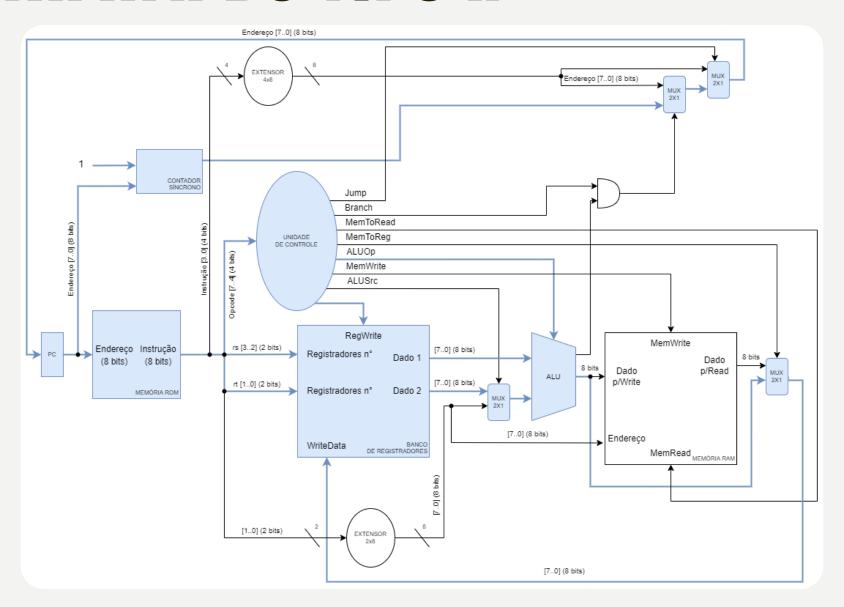
LISTA DE OPERAÇÕES SUPORTADAS

OPCODES	SINTAXE	NOME	EXEMPLO
0000	ADD	SOMA	ADD \$S0 \$S1
0001	ADDI	SOMA IMEDIATA	ADDI \$SO 2
0010	SUB	SUBTRAÇÃO	SUB \$S3 \$S2
0011	SUBI	SUBTRAÇÃO IMEDIATA	SUBI \$S2 4
0100	LW	LOAD WORD	LW \$S0 ENDEREÇO
0101	SW	STORE WORD	SW \$SI ENDEREÇO
0110	LI	LOAD IMEDIATO	LI \$S0 I
0111	BEQ	BRANCH EQUAL	BEQ ENDEREÇO
1000	J	JUMP	J ENDEREÇO
1001	IF	IF	IF \$S1 \$S3

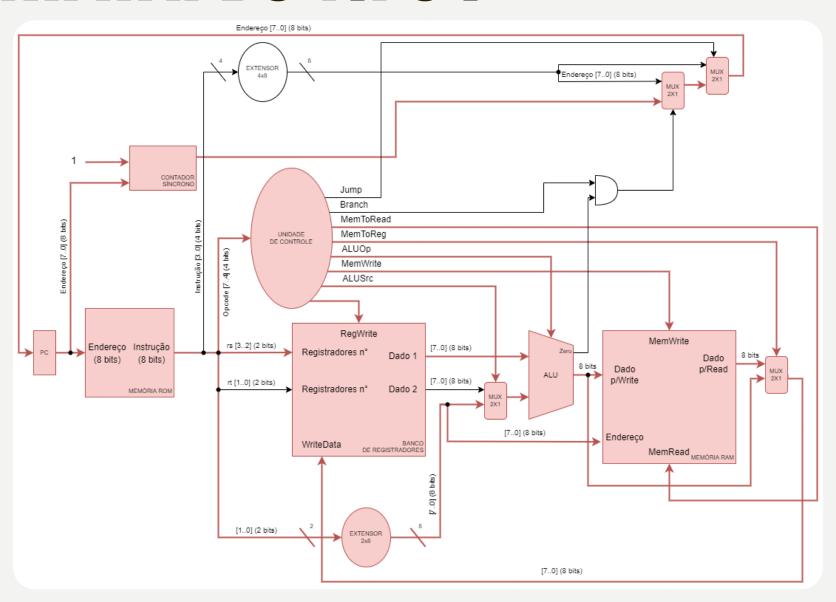
DATAPATH DO PROCESSADOR



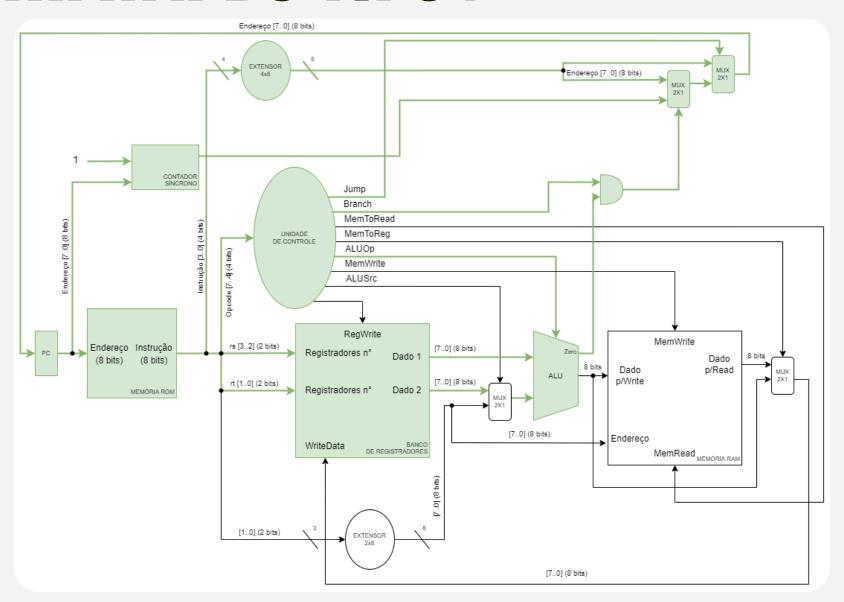
DATAPATH DO TIPO R



DATAPATH DO TIPO I



DATAPATH DO TIPO J



LIMITAÇÕES

- 256 linhas de código em um programa;
- 8 bytes de espaço na memória RAM;
- 4 registradores disponíveis no banco de registradores;
- Jumps apenas podem pular entre as 16 primeiras linhas de código.

TESTE DO ADDI, SUB E SUBI

```
-- TESTE DE ADDI, SUB E SUBI

0 => "00010011", -- ADDI S0 3

1 => "00010101", -- ADDI S1 1

2 => "00110001", -- SUBI S0 1

3 => "00100001", -- SUB S0 S1
```

				III				
-		CLOCK	B 0					
5	>	ALU_RESULT_OUT	B 0000011	00000011	00000001	0000	0010	00000001
ıt.		ALU_OVERFLOW_OUT	B 0					
*	>	ADDRESS_OUT	B 0011	0011	0101	X	000	1
**	>	MUX_2_OUT	B 00000011	00000011	00000001	0000	0010	00000001
3	>	OPCODE_OUT	B 0001		0001	00	11	0010
*	>	PC_OUT	B 00000000	00000000	00000001	0000	0010	00000011
*	>	R_A_OUT	B 00000000	00	000000	0000	0011	00000010
*	>	R_B_OUT	B 00000000	00	000000	X	00000	001
*	>	RAM_OUT	B 00000000					
#	>	ROM_OUT	B 00010011	00010011	00010101	0011	0001	00100001
*	>	RS_OUT	B 00	00	01	X	00	
4	>	RT_OUT	B 11	11 X		0	1	

TESTE DO FIBONACCI PT.1

```
-- TESTE FIBONACCI

0 => "00010000", -- ADDI SO 0

1 => "01010000", -- SW SO

2 => "00010001", -- ADDI SO 1

3 => "00010101", -- ADDI S1 1

4 => "01001100", -- LW S3 00

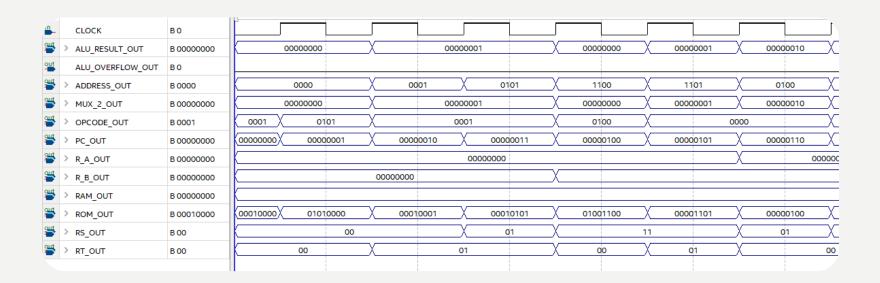
5 => "00001101", -- ADD S3 S1

6 => "00000100", -- ADD S2 S1

7 => "01000000", -- LW SO 00

8 => "0000011", -- ADD S0 S3

9 => "10010100", -- J 0100
```



TESTE DO FIBONACCI PT.2

Overflow

in_	CLOCK	во											
*	> ALU_RESULT_OUT	B 00000000	001	00000000	01011001	10010000	11101001	01011001	10010000	00000000	10010000	11101001	01111001
out	ALU_OVERFLOW_OUT	ВО											
*	> ADDRESS_OUT	B 0000	1	0100	1100	1101	0100	0000	0011	O100 X	1100	1101	0100
*	MUX_2_OUT	B 00000000	001	00	000000	10010000	11101001	00000000	10010000	00000	0000	11101001	01111001
*	> OPCODE_OUT	B 0001	0	1001	0100	00	000	0100	0000	(1001)	01/00	χο	0000
*	> PC_OUT	B 00000000	000	00001001	00000100	00000101	00000110	00000111	00001000	00001001	00000100	00000101	00000110
*	R_A_OUT	B 00000000	000	10010000	01011001	00000000	10010000	01011001	00000000	11101001	10010000	00000000	11101001
*	> R_B_OUT	B 00000000		01011001		10010000	0101	1001	X	10010000		11101001	100100
*	> RAM_OUT	B 00000000											
*	> ROM_OUT	B 00010000	011	10010100	01001100	00001101	00000100	01000000	00000011	10010100	01001100	00001101	00000100
*	> RS_OUT	B 00		V 01	X 1	11	01	X0	10	√X		11	X 01 X
*	> RT_OUT	B 00		X	00	Q1	Χ (00	11	00)	Q1	00

CONCLUSÃO

- Dificuldades encontradas:
 - Criação da memória ROM;
 - Desenvolver soluções em 8 bits;
- Obrigado pela atenção.