Universidade Federal do Rio Grande do Norte
Centro de Tecnologia - CT
Departamento de Engenharia Elétrica - DEE

Disciplina:	ELE1717 - Sistemas Digitais	**Período:**	2018.1
Aluno:		**Data:**	20/03/2018

Implemente o circuito de um microprocessador (Figura 1) com arquitetura do tipo Von-Neumann (RAM 256x8-Bits), com 4 registradores de uso geral (A,B,C,D) de 8 Bits cada, com um ponteiro para pilha (SP) de 8 Bits (na configuração SP-1 com SP máximo no endereço 231), com 2 Bits de sinalização Z (True se o resultado da ULA for zero) e C (True se a ULA apresentar um carry de saída), com 24 posições de I/O representadas pelos endereços de 232 até 255 da memória RAM e com um conjunto de instruções apresentados na Tabela 1.

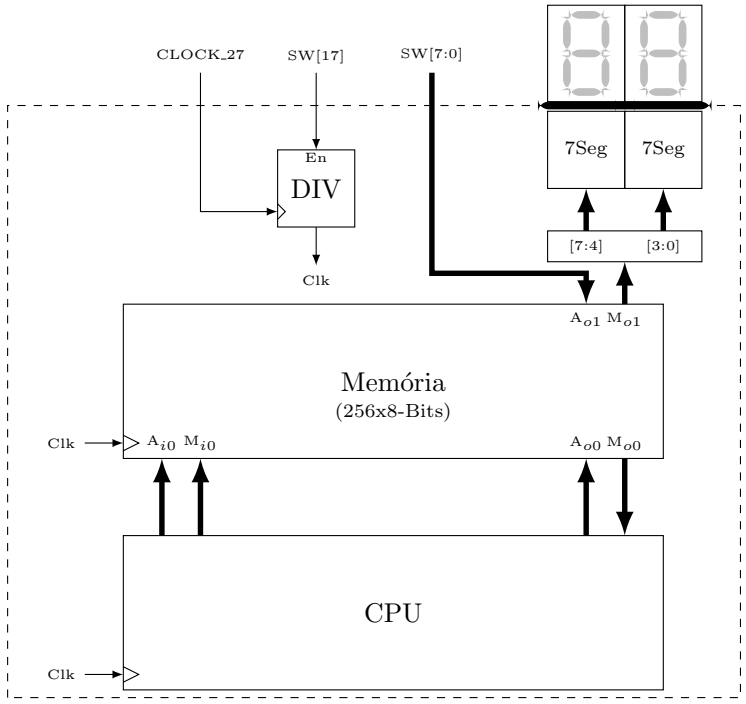

Figura 1: Diagrama de blocos do *datapath* do microprocessador

O processador será construído por um conjunto de blocos constituídos por: **Contador de programa (PC)**, o qual se trata de um registrador que retém o endereço da instrução a ser executada. **Ponteiro da pilha (SP)**, o qual se trata de um registador que retém o endereço do topo da pilha. Quatro **Registradores de uso geral**, o qual se trata de um banco de registadores que retém os dados que estão sendo trabalhados. **Unidade lógica aritmética (ULA)**, o qual se trata de um bloco que realiza operações lógicas e/ou aritmética como soma, subtração entre outras. **Registrador de instrução (IR)**, o qual se trata de um registrador que retém a instrução corrente enquanto ela é executada. O processador possui instruções de três tipos, uma com dois operandos, uma outra com apenas um operando e a última sem a utilização de operandos.

7	6	5	4	3	2	1	0
opcode							
operando 1							
operando 2							

7	6	5	4	3	2	1	0
opcode							
operando 1							

7	6	5	4	3	2	1	0
opcode							

-	**Descrição (2 operandos)**
opcode	Código da função a ser executada
operando 1	Valor a ser utilizado na operação
operando 2	Valor a ser utilizado na operação

-	**Descrição (1 operando)**
opcode	Código da função a ser executada
operando 1	Valor a ser utilizado na operação

-	**Descrição (sem operando)**
opcode	Código da função a ser executada

No processador proposto os dados possuem 8 Bits e o endereçamento da memória 8 Bits. A memória deverá possuir pelo menos uma entrada de dados e duas saídas de dados. A primeira saída de dados (A_{o0},M_{o0}) será destinada ao microprocessador e a segunda (A_{o1},M_{o1}) será utilizada para a verificação do funcionamento do microprocessador. A saída destinada para a verificação terá os seus 8 Bits divididos em 2 grupos de 4 Bits, os quais deverão ser exibidos nos displays de 7-segmentos HEX1,0 através do conversor 7Seg. O microprocessador será implementado no kit DE2, o sinal de *clock* será oriundo do relógio de 27MHz, o sinal de *clock* do sistema deverá ser atribuído ao LEDG[0], a chave SW[17] funcionará como um habilitador do sinal de *clock*, a saída da ULA deverá ser direcionada para os LEDR[15:0] e o botão KEY[1] será utilizado para forçar o *reset* do sistema.

1 (1,0) - Projete um processador, baseado em RTL, para atender as especificações solicitadas (apresentação 27/03/2018).

2 (1,0) - Implemente o datapath do processador (apresentação 03/04/2018).

3 (0,5) - Implemente o bloco de controle do processador (apresentação 03/04/2018).

4 (2,5) - Implemente o processador no Kit DE2 da Altera e faça o relatório do projeto (apresentação 10/04/2018).

Id	Instrução	Descrição
1	MOV reg, regX	reg=regX
2	MOV reg, [regX]	reg=mem[regX]
3	MOV reg, [address]	reg=mem[address]
4	MOV reg, constant	reg=constant
5	MOV [reg], regX	mem[reg]=regX
6	MOV [address], reg	mem[address]=reg
7	MOV [address], Constant	mem[address]=constant
8	ADD reg, regX	reg=reg+regX
9	ADD reg, [regX]	reg=reg+mem[regX]
10	ADD reg, [address]	reg=reg+mem[address]
11	ADD reg, constant	reg=reg+constant
12	SUB reg, regX	reg=reg-regX
13	SUB reg, [regX]	reg=reg-mem[regX]
14	SUB reg, [address]	reg=reg-mem[address]
15	SUB reg, constant	reg=reg-constant
16	MUL reg	[A]=[A]*reg
17	MUL [reg]	[A]=[A]*mem[reg]
18	MUL address	[A]=[A]*mem[address]
19	MUL constant	[A]=[A]*constant
20	AND reg, regX	reg=reg AND regX
21	AND reg, [regX]	reg=reg AND mem[regX]
22	AND reg, [address]	reg=reg AND mem[address]
23	AND reg, constant	reg=reg AND constant
24	OR reg, regX	reg=reg OR regX
25	OR reg, [regX]	reg=reg OR mem[regX]
26	OR reg, [address]	reg=reg OR mem[address]
27	OR reg, constant	reg=reg OR constant
28	XOR reg, regX	reg=reg XOR regX
29	XOR reg, [regX]	reg=reg XOR mem[regX]
30	XOR reg, [address]	reg=reg XOR mem[address]
31	XOR reg, constant	reg=reg XOR constant
32	SHL reg, regX	reg=reg$\ll$regX
33	SHL reg, [regX]	reg=reg$\ll$mem[regX]
34	SHL reg, [address]	reg=reg$\ll$mem[address]
35	SHL reg, constant	reg=reg$\ll$constant
36	SHR reg, regX	reg=reg$\gg$regX
37	SHR reg, [regX]	reg=reg$\gg$mem[regX]
38	SHR reg, [address]	reg=reg$\gg$mem[address]
39	SHR reg, constant	reg=reg$\gg$constant
40	CMP reg, regX	if reg==regX, Z=1
41	CMP reg, [regX]	if reg==mem[regX], Z=1
42	CMP reg, [address]	if reg==mem[address], Z=1
43	CMP reg, constant	if reg==constant, Z=1
44	INC reg	reg=reg+1
45	DEC reg	reg=reg-1
46	NOT reg	reg=not(reg)
47	CALL address	[SP]=PC; SP=SP-1; PC=mem[address]
48	PUSH reg	[SP]=reg; SP=SP-1
49	PUSH [reg]	[SP]=mem[reg]; SP=SP-1
50	PUSH address	[SP]=mem[address]; SP=SP-1
51	PUSH constant	[SP]=constant; SP=SP-1
52	POP reg	reg=[SP]; SP=SP+1
53	JMP address	PC=mem[address]
54	JC	PC=mem[address], if carry=true
55	JNC	PC=mem[address], if carry=false
56	JZ	PC=mem[address], if zero=true
57	JNZ	PC=mem[address], if zero=false
58	JA	PC=mem[address], if carry=false e zero=false
59	JNBE	PC=mem[address], if carry=false e zero=false
60	JAE	PC=mem[address], if carry=false
61	JNB	PC=mem[address], if carry=false
62	JB	PC=mem[address], if carry=true
63	JNAE	PC=mem[address], if carry=true
64	JNBE	PC=mem[address], if carry=true ou zero=true
65	JNA	PC=mem[address], if carry=true ou zero=true
66	JE	PC=mem[address], if zero=true
67	JNE	PC=mem[address], if zero=false
68	RET	PC=[SP]; SP=SP+1
69	HLT	parar o processador

Table 1: Conjunto de instruções do processador.