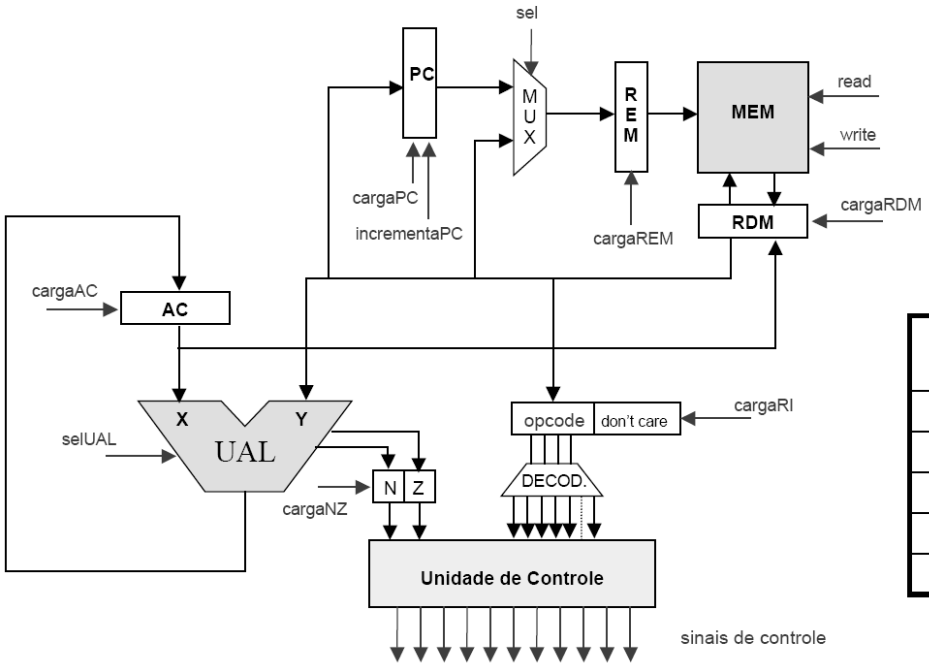
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
DISCIPLINA: TÉCNICAS DIGITAIS

Projeto do Processador Neander em: VHDL

O computador NEANDER foi criado com intenções didáticas[1]. O objetivo deste trabalho é implementar o NEANDER usando portas lógicas básicas **(NAND, NOR, NOT, LATCHES E FLIP_FLOPS)** e simular esse circuito em um simulador lógico de portas para verificar o funcionamento do circuito ao realizar um pequeno programa de 10 instruções gravado na memória.

O computador NEANDER tem as seguintes características:
• Largura de dados e endereços de 8 bits
• Dados representados em complemento de dois
• 1 acumulador de 8 bits (AC)
• 1 apontador de programa de 8 bits (PC)
• 1 registrador de estado com 2 códigos de condição: negativo (N) e zero (Z)



operações da UAL	selUAL
X + Y	000
X and Y	001
X or Y	010
Not X	011
Y	100

[1] Esta pseudo-máquina foi criada pelos Profs. Raul Weber e Taisy Weber para a disciplina Arquitetura de Computadores I da UFRGS.

Projeto do Datapath

Passo 1: Projeto dos circuitos combinacionais

A) Multiplexador 2:1 de largura de 8 bits.

B) Unidade Aritmética e Lógica (UAL): conforme a seleção da UAL (selUAL), 5 operações diferentes podem ocorrer na UAL. A largura dos dados é de 8 bits. Note que a UAL é capaz de identificar quando o resultado é ZERO (Z) ou NEGATIVO (N).

C) Decodificador de instruções: na tabela a seguir AC é o acumulador, MEM(end) significa conteúdo da posição end de memória, N e Z são os códigos de condição e ← representa uma atribuição.

Instrução	Comentário
NOP	nenhuma operação
STA end	MEM(end) ← AC
LDA end	AC← MEM(end)
ADD end	AC← MEM(end) + AC
OR end	AC← MEM(end) OR AC
AND end	AC← MEM(end) AND AC
NOT	AC← NOT AC
JMP end	PC← end
JN end	IF N=1 THEN PC ← end
JZ end	IF Z=1 THEN PC ← end

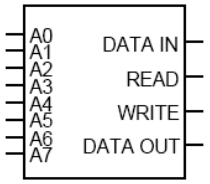
Código	Instrução	Comentário
0000	NOP	nenhuma operação
0001	STA end	armazena acumulador - (store)
0010	LDA end	carrega acumulador - (load)
0011	ADD end	soma
0100	OR end	"ou" lógico
0101	AND end	"e" lógico
0110	NOT	inverte (complementa) acumulador
1000	JMP end	desvio incondicional - (jump)
1001	JN end	desvio condicional - (jump on negative)
1010	JZ end	desvio condicional - (jump on zero)
1111	HLT	término de execução - (halt)

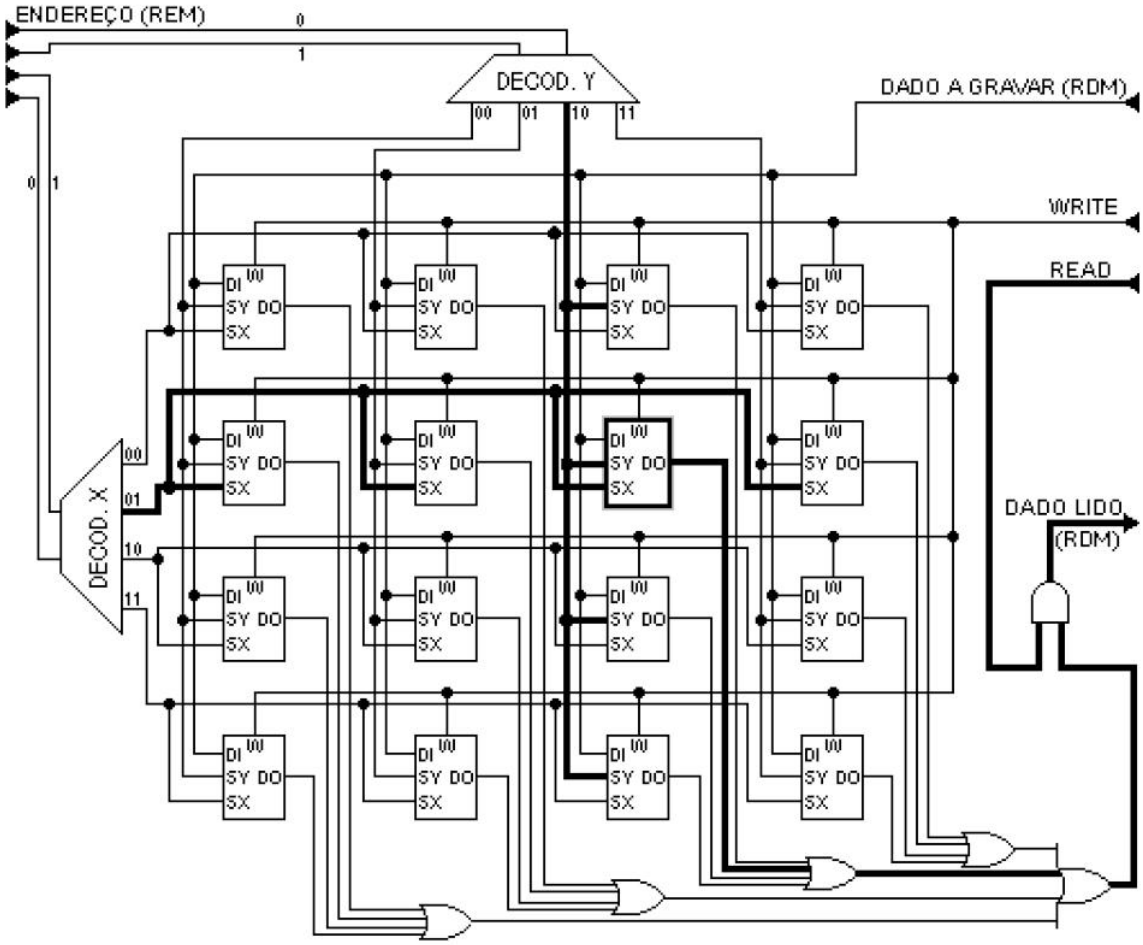
Passo 2: Projeto dos circuitos sequenciais

A) Registradores de 8-bits **ACC, REM, RDM e INST(opcode)** com carga paralela. Notem que todos esses registradores são iguais. Registrador **NZ** de 2 bits com carga paralela. *Onde N - (negativo) : indica sinal do resultado, 1 - resultado é negativo e 0 - resultado é positivo. Z - (zero) : indica resultado igual a zero, 1 - resultado é igual a zero e 0 - resultado é diferente de zero.*

B) Contador de 8-bits **PC** com carga paralela e sinal de incremento.

C) Memória **RAM** para programa e dados. A memoria será um grande banco de registradores de 16 endereços com largura de dados de 8bits. Na memoria haverão registradores, decodificadores de endereços e o seletores. *Logo, neste caso teremos apenas A0, A1, A2 e A3 como entrada de endereço.*





Passo 3: Projeto da Unidade de Controle

A unidade de controle é uma maquina de estados finita (FSM) que controla a leitura e escrita da memoria e os elementos do Datapath conforme os sinais do decodificador de instrução e a temporização do processador.

Passo 4: Projeto do programa de 10 instruções

A memória projetada ao ser inicializada com o sinal de RESET deve conter um programa inicial de 10 instruções dos endereços 0000 a 1010. Os 6 endereços restantes da memória devem ser usados para dados do programa (variaveis a serem utilizadas). Cada grupo deverá criar um programa diferente.