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

Unidade Central de Processamento
ELE1717 - Grupo 02 - Problema 02 - Projeto

Ana Beatriz Marinho Neves
Elias Gurgel de Oliveira
Isaac de Lyra Junior
Wesley Brito da Silva

Natal, 29 de junho de 2021

Resumo

Neste relatório é proposto um modelo para a criação de um circuito integrado (CI), o qual consiste em um processador programável de propósito geral. Para tanto, foi utilizado o método RTL no desenvolvimento do mesmo, de forma que o controlador do bloco de dados e das memorias consiste em uma máquina de estados finitos. O modelo do processador proposto é baseado na arquitetura Harvard, onde a memória de dados é distinta da memória de instruções. Ao final deste, conforme toda teoria e esquemáticos apresentados, é possível construir o circuito proposto que pode ser usado de acordo com as suas instruções de operação.

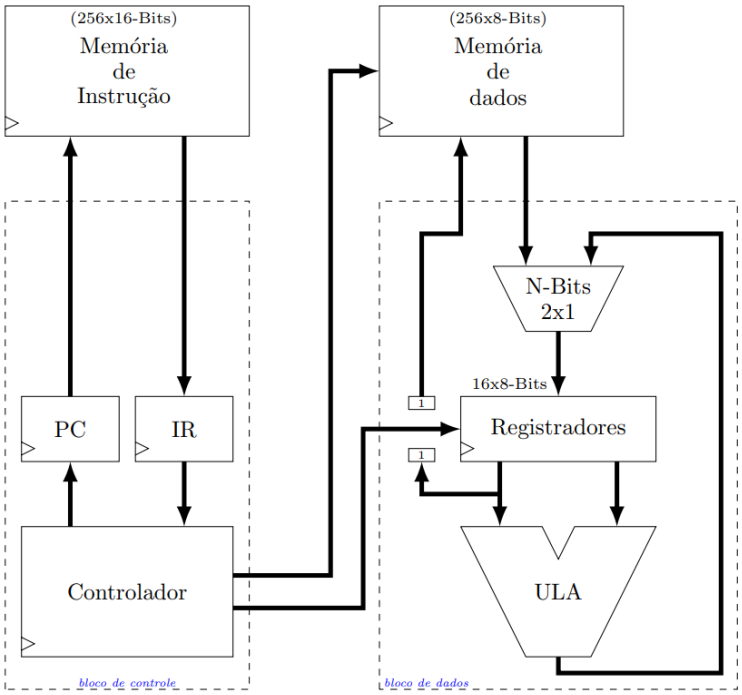
Palavras-chave: Sistemas Digitais, Processador, Memória, Registradores de propósito geral, ULA.

Sumário

1 INTRODUÇÃO

O projeto dessa semana é a elaboração de um CI, que deverá possuir a estrutura da Figura 1, desenvolvido para a implementação de uma Unidade Central de Processamento (CPU) que possui 16 instruções, de acordo com a Figura 2.

Figura 1 – Estrutura básica do projeto RTL de uma CPU de uso geral



Fonte: Dados do problema.

Figura 2 – Conjunto de instruções da CPU de uso geral

Oper.	Classe	Opcode	4bits	4bits	4bits	Descrição	Carry	ULA
HLT	Controle	0000	-	-	-	PC_{k+1}=PC_k		
LDR	Dados	0001	A	addr[7..4]	addr[3..0]	Reg[A]<=Mem_D[addr]		
STR	Dados	0010	A	addr[7..4]	addr[3..0]	Mem_D[addr]<=Reg[A]		
MOV	Dados	0011	-	B	C	Reg[B]<=Reg[C]		
ADD	ULA	0100	A	B	C	Reg[A]<=Reg[B]+Reg[C]	•	•
SUB	ULA	0101	A	B	C	Reg[A]<=Reg[B]-Reg[C]	•	•
AND	ULA	0110	A	B	C	Reg[A]<=Reg[B] AND Reg[C]	•	•
OR	ULA	0111	A	B	C	Reg[A]<=Reg[B] OR Reg[C]	•	•
NOT	ULA	1000	A	-	C	Reg[A]<=NOT Reg[C]	•	•
XOR	ULA	1001	A	B	C	Reg[A]<=Reg[B] XOR Reg[C]	•	•
CMP	ULA	1010	A	B	C	Reg[A]<=CMP(Reg[B],Reg[C])	•	•
JMP	Salto	1011	-	value[7..4]	value[3..0]	PC_{k+1}=value		
JNC	Salto	1100	-	value[7..4]	value[3..0]	PC_{k+1}=value, if carry=0		
JC	Salto	1101	-	value[7..4]	value[3..0]	PC_{k+1}=value, if carry=1		
JNZ	Salto	1110	-	value[7..4]	value[3..0]	PC_{k+1}=value, if ULA≠0		
JZ	Salto	1111	-	value[7..4]	value[3..0]	PC_{k+1}=value, if ULA=0		

Fonte: Dados do problema.

A sequência de passos a serem executados pelo processador programável estará pré-definido na memória de instruções. Esta, por sua vez, possui um barramento de endereços de 8 *bits*, o que resulta em 256 registros de memória, e um barramento de 16

bits de instruções por registro. Neste projeto o barramento de endereços é alimentado com uma palavra binária fornecida pelo contador de programa (PC).

Após definido o endereço da instrução a ser executada, a memória envia os 16 *bits* correspondentes a esse endereço para o registrador de instruções (IR), responsável por alimentar os estágios seguintes com essas informações.

Na memória de dados estarão disponíveis os valores que poderão alimentar o banco de registradores através de um MUX, ela também pode receber e gravar valores vindos do banco de registradores. O MUX é responsável por escolher o dado que será gravado no banco de registradores de arquivos (RF). Entre suas opções estão os dados vindos da memória ou os dados vindos do resultado de alguma operação realizada pela ULA. Por fim, a ULA será a responsável pelas operações aritméticas e booleanas da máquina.

No decorrer deste relatório, serão descritos em detalhes, cada etapa do projeto RTL no desenvolvimento deste projeto.

2 DESENVOLVIMENTO

2.1 Arquitetura Básica

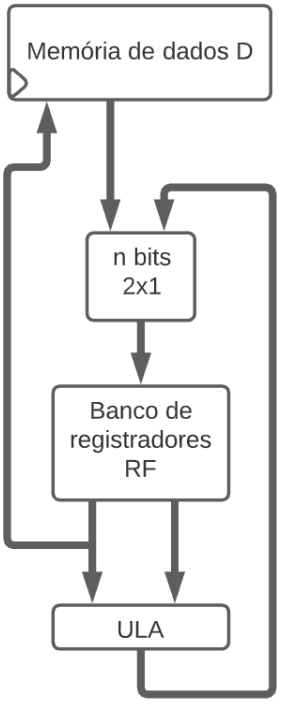
Um processador programável consiste em duas partes principais: um bloco operacional (*datapath*) e uma unidade de controle. No bloco operacional básico, a carga de dados consiste na execução da leitura dos dados, com os quais se quer trabalhar, em algum local de entrada. A transformação desses dados consiste em realizar algumas computações com os mesmos, de forma que o resultando tem por objeto novos dados. Por fim, o armazenamento dos novos dados, significa escrever os mesmos em algum local de saída (VAHID, 2008).

2.1.1 Bloco operacional básico

A memória de dados consiste em um dispositivo de entrada e saída. Nela, são armazenados valores que serão utilizados pelo processador programável. Além disso, essa memória também serve para armazenar o produto das operações do processador. Durante o processamento dos dados, um processador programável precisa ser capaz de carregar os dados da memória em um banco de registradores, que estão localizados dentro dele mesmo. Ele precisa ainda ser capaz de alimentar as unidades funcionais que, a partir de um subconjunto dos registradores, podem executar todas as operações de transformação que irão ser cogitadas (comumente em uma ULA) (VAHID, 2008).

Os resultados das operações de ULA são armazenados no banco de registradores de propósito geral (RF) e, após essa ação, eles estarão disponíveis para serem armazenados na memória de dados. Em outras palavras, a ULA opera apenas com os dados que estão disponíveis no banco de registradores de propósito geral (RF), além disso, a memória de dados faz operações de envio e recebimento de valores, apenas com o banco de registradores de propósito geral (RF). Todo essa descrição é para mostrar a importância do banco de registradores no processamento dos dados efetuados pela ULA. Esse conjunto de elementos executa um papel de bloco intermediador entre a ULA e a memória de dados, conforme demonstrado anteriormente. Um fato importante a se observar é que tanto a memória de dados quanto o banco de registradores são endereçados pelo bloco de controle da CPU, mostrando que essas duas partes devem estar em perfeita sincronia para que todo o processamento de dados ocorra sem erros ou falhas. Todo esse circuito descrito é conhecido como bloco operacional (*datapath*) do processador (VAHID, 2008). Na Figura 3 podemos ver ilustrado um bloco operacional.

Figura 3 – Bloco operacional de um processador.



Fonte: VAHID, 2008 [ADAPTADO].

2.1.2 Unidade de controle básica

A unidade de controle lê, na "memória de instruções", a instrução a ser realizada e então a executa no bloco operacional. Esse processo é dividido em três estágios conhecidos como ciclo de máquina, onde cada estágio é responsável por um conjunto de tarefas específicas. Os estágios mencionados anteriormente são: busca, decodificação e execução.

No primeiro estágio (BUSCA) as instruções contidas no endereço apontado por PC são carregadas no registrador de instruções (IR), após isso, o contador de programa (PC) é incrementado concluindo o estágio de busca. No segundo estágio (DECODIFICAR) a informação contida no IR é decodificada (compreendida) e os blocos internos do circuito são orientados a configurar os sinais binários para o estágio seguinte, de acordo com o comando a ser realizado. Por fim, a operação é executada, ativando, se for o caso, as linhas de controle apropriadas para o bloco operacional (VAHID, 2008).

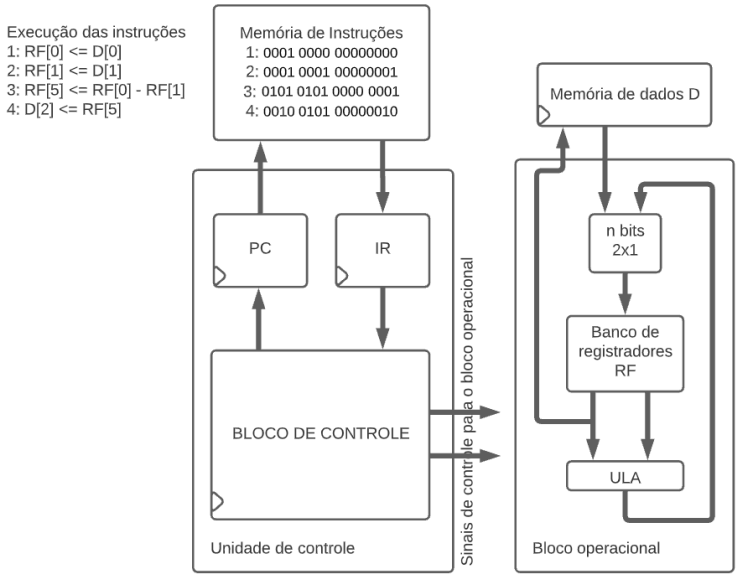
2.1.3 Comunicação entre o bloco operacional e unidade de controle

A comunicação entre o bloco operacional e unidade de controle se dará por meio de três tipos de barramentos: o de controle, por onde são enviados, em via dupla, sinais de controle entre os dois blocos. O barramento de dados, por onde fluem informações como um valor que deve ser armazenado na memória. E por fim, o barramento de endereços, por onde são informados os endereços de memórias e/ou registradores a serem utilizados

em uma operação.

Um exemplo de como ocorre esta comunicação pode ser visto na Figura 4, onde um conjunto de 4 instruções foi previamente armazenado na memória de instruções. Pode ser observado que o bloco de controle realiza cada uma delas de acordo com o ciclo de máquina já descrito anteriormente. O PC fornece o endereço da primeira instrução a ser executada, que no nosso caso, consiste no carregamento do registrador RF[0] com o dado contido no endereço D[0] de memória de dados, em seguida o PC é atualizado. No segundo ciclo o IR é carregado novamente com a nova instrução. Em uma operação semelhante, os dados contidos no endereço de memória D[1] são carregados para o registrador RF[1] e o PC é atualizado, concluindo esse ciclo. No terceiro ciclo o IR é carregado novamente com a instrução contida no endereço de memória apontado por PC e o mesmo é atualizado. Depois da decodificação, para saber de que operação se trata, a máquina configura todos os sinais para executar, no passo seguinte, uma operação de subtração entre os valores contidos nos registradores RF[0] e RF[1], e armazena o resultado desta operação no registrador RF[5]. Na execução da última instrução, o bloco de controle emitirá os sinais para que os dados inclusos no registrador RF[5] sejam armazenados no endereço D[2] de memória.

Figura 4 – Comunicação entre o bloco operacional e unidade de controle



Fonte: VAHID, 2008 [ADAPTADO]

3 PROJETO

Para o desenvolvimento do projeto da CPU, utilizamos o passo-a-passo de um projeto RTL onde começamos pela MDE de alto nível para, posteriormente, montamos a estrutura da CPU de uso geral com o clock de 1 GHz, o bloco de controle e o bloco operacional, e por fim organizamos a tabela verdade com as variáveis da MDE de baixo nível.

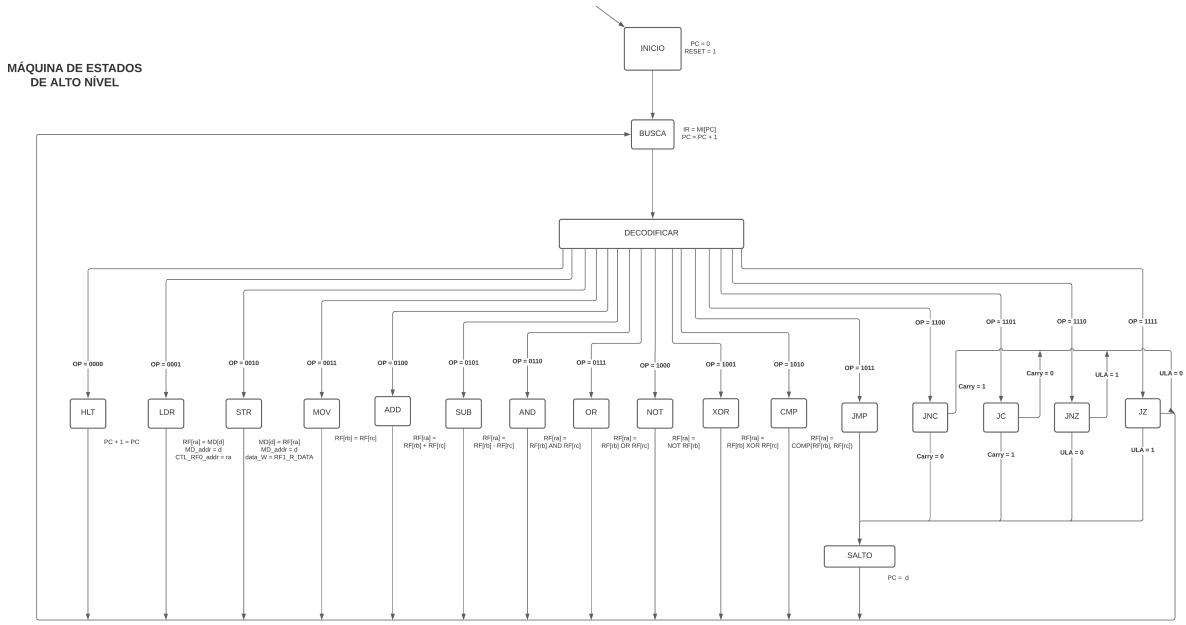
3.1 Máquina de Estados

No início do projeto, começamos a elaborar uma máquina de estados (MDE) capaz de atender as especificações definidas no PDF do projeto.

3.1.1 Alto Nível

Primeiramente, foi implementado uma MDE de alto nível e sua estrutura está definida na Figura 5.

Figura 5 – Máquina de estados de alto nível



Fonte: Elaborado pelos autores.

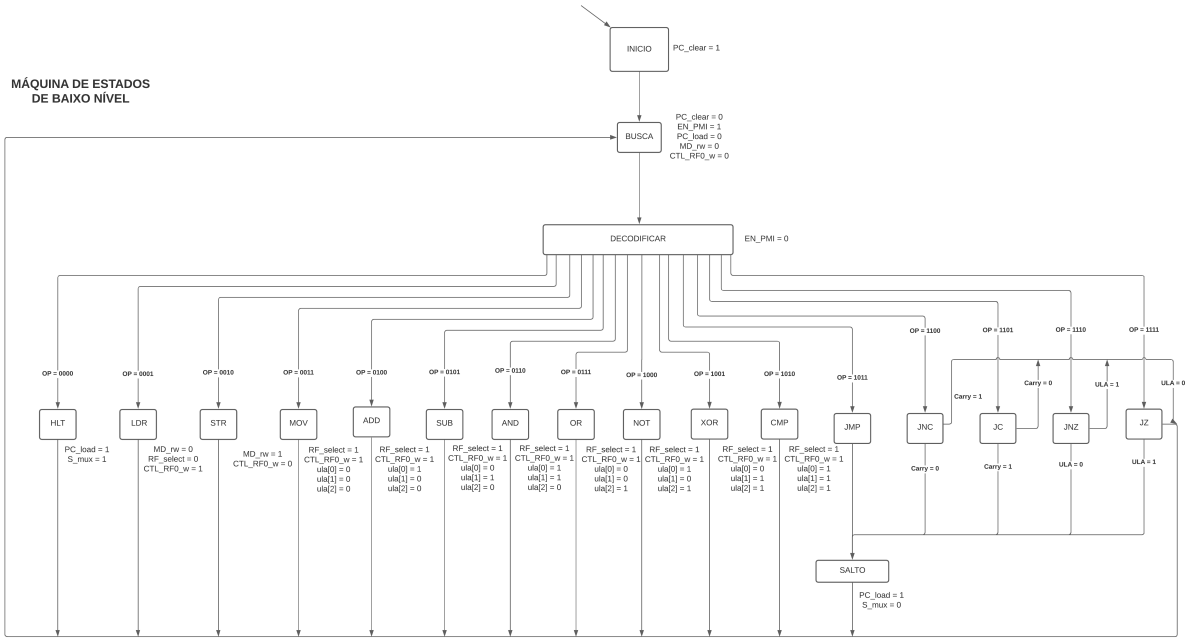
A MDE de alto nível possui 20 estados, sendo eles: INÍCIO, BUSCA, DECODIFICAR e SALTAR, os outros 16 estados correspondem as instruções pré-definidas no projeto. No estado INÍCIO o PC começa apontando para o endereço '0' da memória de instruções,

no estado seguinte (BUSCA) o IR receberá a informação contida no endereço de memória de instrução apontada pelo PC e o PC é incrementado. No estado de DECODIFICAR será verificado o valor do *opcode* e ele definirá o estado futuro, ou seja, uma das 16 instruções fundamentadas no projeto. Caso o *opcode* consista em uma instrução que realize um salto, no estado de execução o PC é atualizado com os 8 últimos *bits* constantes no registrador IR, que correspondem a um endereço na memória de instruções, em caso de saltos. Após isso a máquina retorna ao estado de BUSCA, reiniciando um novo ciclo de máquina.

3.1.2 Baixo Nível

Na MDE de baixo nível, foram acrescentadas as saídas de cada estado, como mostra a Figura 6. Fora isso, não há modificação entre a MDE de alto nível para da MDE de baixo nível.

Figura 6 – Máquina de estados de baixo nível



Fonte: Elaborado pelos autores.

3.2 Bloco de Controle

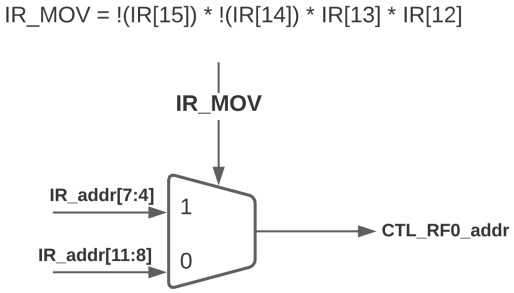
O bloco de controle possui as entradas: CARRY, ULA, OP(3), OP(2), OP(1) e OP(0), como apresenta a tabela verdade no Anexo B.

As entradas CARRY e ULA diferente do *opcode*, são saídas da operação que ocorrerá dentro da ULA. Já o *opcode* consiste nos 4 *bits* mais significativos armazenados no IR.

3.3 Bloco operacional

O bloco operacional possui dois MUX 2x1 que endereçará os registradores em dois casos específicos. O primeiro caso é o da instrução MOV onde o primeiro MUX ao invés de pegar o endereço [11:8], ou registrador A (ra), ele enviará o endereço [7:4] que corresponde ao registrador B (rb) que escreverá no registrador RF0, como pode ser visto na Figura 7.

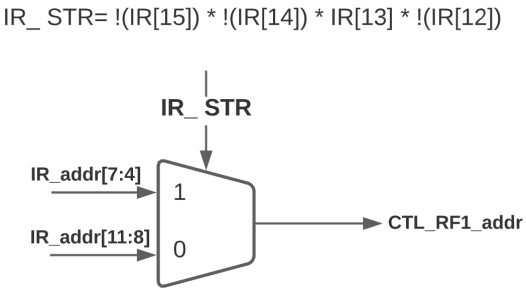
Figura 7 – MUX que define o endereço do registrador de escrita



Fonte: Elaborado pelos autores.

O segundo caso é o da instrução STR, onde o endereço do rb passa para o ra, como mostra a Figura 8.

Figura 8 – MUX que define o primeiro endereço do registrador de leitura



Fonte: Elaborado pelos autores.

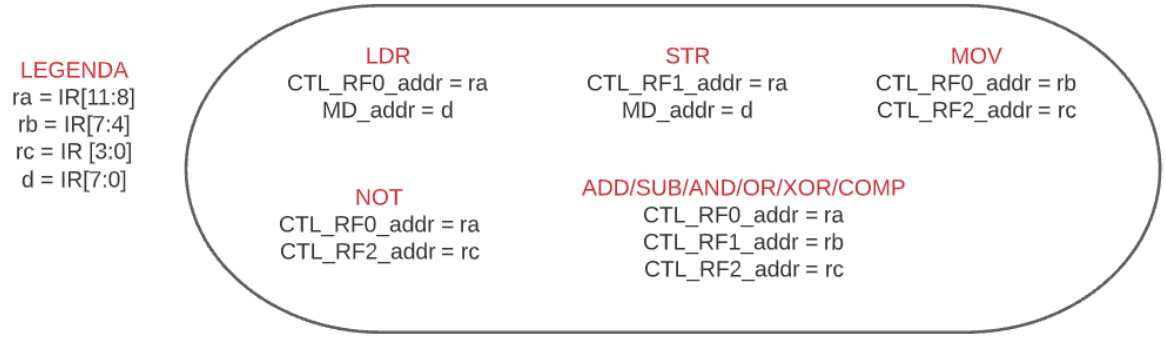
A Figura 9 mostra uma relação de endereçamento que o bloco de controle deve realizar a depender do *opcode* recebido pelo registrador de instrução.

A Figura 10 mostra a estrutura completa do bloco operacional pensado durante a fase de projeto, nas seções a seguir iremos discutir um pouco de cada componente.

3.3.1 Endereços

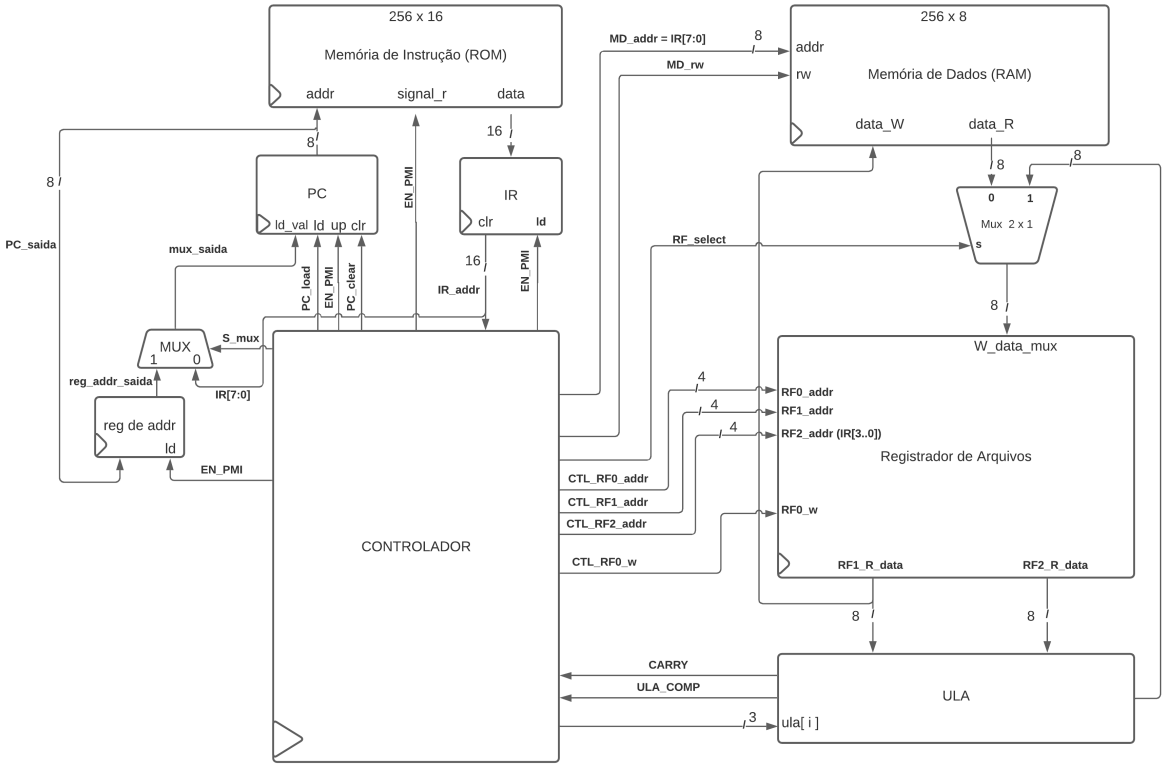
Considerando a instrução de 16 *bits* contida no IR, de acordo com este projeto, podemos segmentá-la em 4 partes: IR[15..12] consiste no *opcode*; IR[11..8], a depender do *opcode*, pode significar o endereçamento do registrador onde será gravado alguma informação, dentro do banco de registradores, ou não ter utilidade; IR[7..4], a depender do *opcode*, pode significar um endereço de leitura para o banco de registradores, uma

Figura 9 – Relação de endereçamento



Fonte: Elaborado pelos autores.

Figura 10 – Máquina completa



Fonte: Elaborado pelos autores.

parte de um endereço da memória de dados ou uma parte de um endereço da memória de instruções; Por último, IR[3..0], também considerando o *opcode*, pode significar o endereço de leitura no banco de registradores, uma parte do endereço da memória de dados ou uma parte do endereço da memória de instruções.

3.3.2 Memórias

Neste projeto serão utilizados duas memórias, uma do tipo *Read Only Memory* (ROM), que é capaz de armazenar até 256 instruções de 16 *bits* dentre as 16 possíveis, distintas e pré-definidas. A outra memória que será utilizada é do tipo *Random Acces Memory* (RAM), essa memória também possui 256 endereços, porém de 8 *bits*. Essa última poderá gravar ou fornecer informações para o banco de registradores.

3.3.2.1 Memória de Instruções

A memória de instruções possui um barramento de endereços de 8 *bits* e é endereçada pelo PC. Além disso ela possui um sinal habilitador de operações de 1 *bit*. Após receber o endereço para leitura e o *bit* que permite as operações, essa memória disponibilizará em seu barramento de dados de 16 *bits*, o conjunto de informações contidas no endereço informado pelo PC.

Essa memória de instruções, como dito anteriormente, será do tipo ROM, ou seja, o usuário não poderá modificar as instruções ali contidas, nem adicionar ou deletar um *opcode* e nem mudar o endereço dos registradores ou da memória de dados.

3.3.2.2 Memória de Dados

A memória de dados possui um barramento de endereços de 8 *bits*, podendo endereçar até 256 registros de 8 *bits*. Possui também um *bit* para habilitar a leitura ou escrita, elá é usada para dar suporte ao processamento dos dados pela ULA, armazenando dados vindos do banco de registradores ou fornecendo dados ao mesmo.

Essa memória, conforme descrito anteriormente, é do tipo RAM, sendo capaz de armazenar ou disponibilizar os dados contidos nos seus registros.

3.3.3 Registradores

Nesse projeto possuímos vários registradores, são eles: reg_addr - é um registrador que contém o endereço anterior de PC, ou seja, um registrador que armazena o valor de PC antes dele ser atualizado; IR - é o registrador que armazena a instrução contida no endereço de memória apontado por PC; O PC - é um registrador que aponta para o endereço de memória onde contém a instrução corrente a ser carregada pelo IR no ciclo de busca. Após a carga do IR, o PC é incrementado de forma a apontar para a instrução seguinte. Dependendo do contexto, o PC também pode ser carregado com outro valor não sequencial, é o caso da execução de uma operação de parada da CPU, quando o PC é carregado a cada ciclo de máquina, com o valor de reg_addr. Nas operações de salto (*jump*), onde o IR deve receber a instrução contida em um endereço específico da memória de instruções. Nesse último caso, o bloco de controle provê toda a sinalização necessária

para que o PC seja atualizado corretamente. Os sinais de trabalho do PC são: PC_load que carrega o PC com os dados contidos na saída do MUX (essa informação pode ser o valor anterior de PC ou um endereço de memória contido em IR[7:0], em caso de salto), PC_clear que põe em zero todos os bits de saída do PC e EN_PMI que faz o PC apenas ser incrementado; RF - também conhecidos como registradores de uso geral, dependendo da literatura, consiste em um conjunto de registradores onde são armazenados os dados a serem processados pela ULA, como também o resultado do processamento da mesma. Nesse projeto, o RF consiste em um conjunto de 16 registradores de 8 *bits*, e terá como entradas: 3 entradas de 4 *bits* vindas dos últimos 12 *bits* de IR que irão fornecer ao bloco operacional o endereço dos registradores em que ocorrerá a escrita e leitura de dados e um sinal vindo do bloco de controle para ativar ou desativar a escrita de dados.

3.3.4 ULA

A ULA utilizada poderá efetuar 8 operações, que neste caso serão: ADD, SUB, AND, OR, NOT, XOR e CMP. Além disso, ela também tem participação instrução MOV, onde ela deverá dar livre passagem aos dados de um dos canais de entrada de forma a fornecê-los em sua saída sem qualquer alteração. Com isso, o banco de registradores consegue transferir o conteúdo de um registrador para outro. Na adição, o barramento de entrada de 3 *bits* receberá o código da determina função vindo do bloco de controle e os dados da operação são advindos do registrador de arquivos através das duas entradas de 8 *bits*. Após isso, o resultado sairá mediante o barramento de 8 *bits* que vai em direção ao MUX 2x1 para que seja armazenado no banco de registradores. Neste momento, RF_select será '1', ativando o a escrita em um registrador. Caso tenha algum *bit* extra como *carry out*, este irá para a saída *carry* .

Todos esses processos também ocorrerão para a subtração, função AND, função OR, função NOT (inversão do valor de apenas um registrador) e a função XOR. Para o COMP consideramos que é uma comparação dos valores de dois registradores. Esta comparação irá efetuar uma subtração entre os valores recebidos, caso o resultado seja '0', os dígitos de 8 *bits* serão iguais, nesse caso a saída ULA_COMP terá valor igual a '1' e a saída que vai direto para o MUX receberá no seu *bit* menos significativo o nível lógico alto. Isto foi pensado com o intuito de reutilizar as estruturas já criadas, evitando a criação de novos arranjos.

Referências

VAHID, F. *Sistemas Digitais: Projeto, Otimização e HDLs*. [S.l.]: Artmed Bookman, 2008.

ANEXO A – Relato semanal

Líder: Ana Beatriz Marinho Neves

A.1 Equipe

Tabela 1 – Identificação da equipe

Função no grupo	**Nome completo do aluno**
Redator	Isaac de Lyra Junior
Debatedor	Elias Gurgel de Oliveira
Videomaker	Wesley Brito da Silva
Auxiliar	-

Fonte: Produzido pelos autores.

A.2 Defina o problema

O projeto dessa semana da disciplina de Sistemas Digitais é a elaboração de um sistema digital que implemente uma CPU de uso geral com 16 instruções. As instruções [Figura 2] foram pré-definidas no PDF disponibilizado pelo professor e a partir do exemplo de uma estrutura geral da CPU [Figura 1] desenvolvemos o nosso projeto.

A.3 Registro de *brainstorming*

Foram realizadas reuniões semanalmente no horário N12, esse era o horário disponível por todos, e durante as reuniões discutíamos as ideias para o projeto assim como as dúvidas que surgiam sobre o funcionamento de alguns blocos.

Na primeira reunião, na terça dia 22/06 após a aula de Sistemas Digitais, definimos a função de cada integrante e combinamos de estudar o Capítulo 8 para podermos discutir o projeto na próxima reunião.

Na segunda reunião, na quarta dia 23/06 no N12, discutimos algumas dúvidas relacionadas ao projeto e sobre o funcionamento da Estrutura geral de uma CPU.

Na terceira reunião, na quinta dia 24/06 no N12, finalizamos a MDE de alto nível e começamos a discussão sobre as saídas do bloco de controle e operacional.

Na quarta reunião, na sexta dia 25/06 no N12, começamos a tabela verdade de projeto, porém não foi finalizada no dia pois ainda não tínhamos terminado o bloco de controle e o operacional.

Na quinta reunião, no sábado dia 26/07 no N12, finalizamos a estrutura do projeto RTL da CPU de uso geral, finalizamos a tabela verdade e começamos a elaborar a MDE de baixo nível.

Na sexta reunião, no domingo dia 27/06 a partir das 16:00 horas, fomos revisar o projeto e identificamos algumas falhas que foram consertadas no mesmo dia. No início da noite, começamos a escrita desse relatório.

A.4 Pontos-chaves

O ponto chave deste relatório foi compreender como funciona a arquitetura básica da CPU, pois alguns integrantes ainda não tinham pago a matéria de Arquitetura de Computadores e estavam com dificuldade de entender como funcionaria as rotinas do projeto.

A.5 Questões de pesquisa

- Projetos RTL;
- Arquitetura básica de uma CPU;
- Funcionamento do Contador de Programa;
- Funcionamento do Registrador de Instruções.

A.6 Planejamento da pesquisa

O livro Sistemas Digitas do Frank Vahid foi a literatura base para o nosso projeto, o livro possui um projeto parecido com o qual iríamos desenvolver então, a maior parte dos estudos foi por ele.

ANEXO B – Tabela de Transição de Estados