UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

LOURIVAL OLIVEIRA DA SILVA

Projeto de Microprocessador Digital

Feira de Santana, 19 de Julho de 2004

UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

LOURIVAL OLIVEIRA DA SILVA

Projeto de Microprocessador

Relatório do Problema de Projeto de Microprocessador Digital apresentado para avaliação da Disciplina de Arquitetura e Organização de Computadores do 2º. semestre, do Curso de Engenharia de Computação, da Universidade Estadual de Feira de Santana sob orientação do Prof. Dr. Márcio.

Feira de Santana, 19 de Julho de 2004

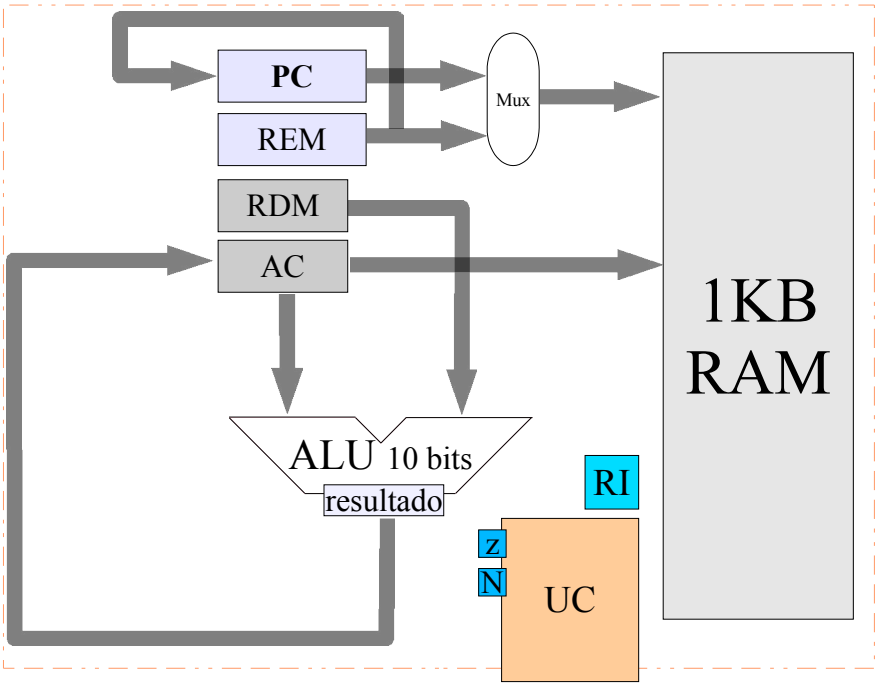
Sumário

Introdução

Discutir o projeto e a implementação de um microprocessador programavél é o intuito deste trabalho. O tema será tratado de uma forma mais abstrata no começo, discutindo sobre o que o processador deve fazer, e as considerações que foram feitas nas tomadas de decisões que levaram à arquitetura proposta, ao longo do texto procurou-se explanar cada detalhe desta arquitetura, definindo-se a função de cada componente empregado e os caminhos de dados que os interligam. Será abordada a implementação de cada registrador, tanto o de propósito geral (AC), como os de propósito espécifico com RDM, e REM. Depois será apresentada a implementação da ULA, com seus sub-componentes e código de controle. Por fim o trabalho busca esclarecer a implementação da Unidade de Controle, detalhando as considerações feitas para definição e geração das microinstruções implementadas, tanto para o processo de busca como também para decodificação e execução de cada função proposta.

O Processador

O projeto de microprocessador proposto apresenta as caracteristicas básicas dos processadores de propósito geral. As suas instruções operam sobre 8 bits de dados, sendo capaz de endereçar 1024B de dados, a memória e compartilhados entre os dados e as instruções do sistema. O diagrama de blocos abaixo proporciona uma visão geral das partes que compõem o sistema.



Programação

Um computador de propósito geral deve ser capaz de executar diferentes sequências de instruções. Para tal estas instruções são lidas de algum tipo de memória não volátil e carregadas na memória para que então possam ser executadas pelo processador, sendo assim para que tais comandos possam ser processados eles devem seguir as normas da arquitetura interna do processador, que neste exemplo foi definida como segue.

As Instruções

Cada instrução é composta por 16 bits onde os 4 primeiros guardam o código da operação, e os próximos dez bits indicam o endereço de memória do operando. Apesar de algumas funções como NOP

e HLT não possuirem operandos seus tamanhos foram mantidos em 16 bits para respeitar a uniformidade das instruções, pois fica mais fácil trabalhar com instruções uniformes, assim dispensa-se a implementação de diferentes instruções de busca, esta mesma observação justifica o fato das intruções terem 16 bits e não os 14 realmente nescessários, este arranjo se justifica pelo fato das memórias trabalharem com palavras de 1 byte, mantendo-se as instruções em 16 bits facilitou a leitura da instrução na memória. Os 16 bits de cada instrução na memória são ocupados de acordo a seguinte conveção:

1º	*2º*	*3º*	*4º*	*5º*	*6º*	*7º*	*8º*	*9º*	*10º*	*11º*	*12º*	*13º*	*14º*	*15º*	*16º*
Código da Operação				Reservado		Endereço de memória do operando									

O código de cada operação foi definido de acordo com a tabela abaixo, observe que os bits dos códigos das instruções ADD, OR, AND e NOT, LDA coincidem com os valores utilizados para controlar a ALU exibidos mais adiante, assim nestas operações basta copiar este bits para entrada de controle de operação da ALU simplificando assim o projeto da UC que será detalhado no final deste trabalho:

Inst	*OPCod*				*Hx*	*1° Operando*	*2° Operando*	*Saida Para*	*Classe*
	2^3	2^2	2^1	2^0					
	B	*A*	F_1	F_0					
STA	0	1	0	1	**0A**	Endereço de Destino	Acumulador	Memória	Mem
LDA	1	0	1	1	**0B**	Endereço de Origem	Acumulador	Acumulador	Mem
ADD	1	1	1	1	**0F**	Endereço de Origem	Acumulador	Acumulador	Mat
OR	1	1	1	0	**0E**	Endereço de Origem	Acumulador	Acumulador	Mat
NOT	1	1	0	1	**0D**	-	Acumulador	Acumulador	Mat
AND	1	1	0	0	**0C**	Endereço de Origem	Acumulador	Acumulador	Mat
JMP	0	0	0	1	**01**	Endereço de Memória	-	PC	Desvio
JN	0	1	0	1	**05**	Endereço de Memória	Reg.Status N	PC	Desvio
JZ	1	0	0	1	**09**	Endereço de Memória	Reg.Status Z	PC	Desvio
NOP	1	0	0	0	**08**	-	-	-	Sistema
HLT	0	0	0	0	**00**	-	-	-	Sistema

Arquitetura do processador

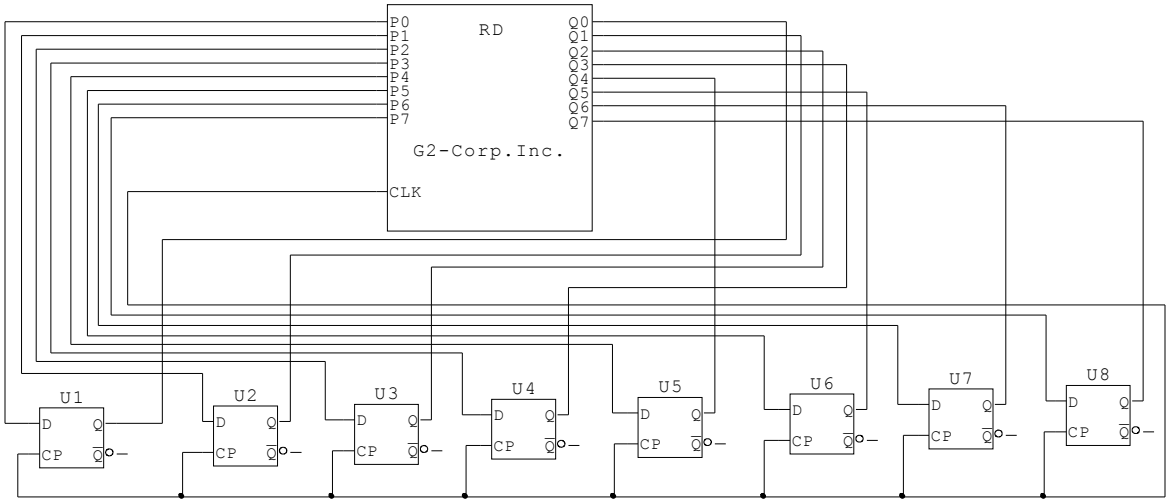
A nivél de hardware este projeto segue a arquitetura proposta nas literaturas, que distinguem um conjunto básico de componentes para um processador, sendo estes os seguintes:

Contador de Programa, Acumulador, Registrador de Dados da Memória, Registrador de Endereço de Memória, Unidade Aritmética Lógica, Registradores de Estado, Unidade de Controle

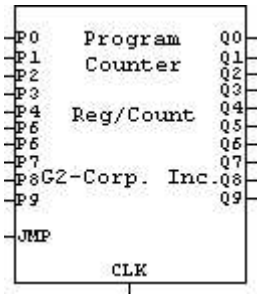
Cada uma destas partes tem funções bem definidas e são ativadas de forma bem controlada pela Unidade de Controle durante a busca e execução de cada instrução. As seções seguintes visam detalhar cada uma destas partes que compõe o projeto proposto.

Registradores

Registradores são as unidades básicas de armazenamento do processador, neste projeto há somente um registrador de uso geral AC, e outros dois de propósito especifico RDM de 8 bits e REM de 10 bits, a implementação de ambos segue o mesmo principio, um conjunto de Flip-Flops tipo D com o sinal de clock associado, para que seja realizada a escrita simultânea de todos os Flip-Flops, o valor a ser escrito é aquele que estiver presente na entrada P0 a P7 durante a transição positiva de clock como definido para este tipo de componente, a figura abaixo expõe a implementação de um registrador:



Contador de Programa (PC)



Este é um registrador de propósito espécifico ele é responsavél por armazenar o endereço da próxima instrução a ser executada no processador, por representar endereços de memória este registrador possui dez bits de capacidade, isto o habilita a endereçar os 1024 bytes de dados da memória do sistema. O valor de PC é incrementado duas vezes durante cada ciclo de execução, a primeira incrementação ocorre após a leitura da operaçõa da memóra, então o operando é copiado da memóra e o PC é incrementado uma segunda vez, assim seu valor passa a apontar para o endereço da próxima instrução a ser executada. Além da modificação por incrementação o endereço de PC pode ser alterado pelas instruções desvio (jz, jn, jmp) que definem um novo valor para este registrador efetuando assim desvios na execução do programa. Para facilitar a sua incrementação o PC é implementado como um conjunto de contadores JK assincronos, assim a incrementação é feita por um simples pulso de clock na entrada CLK incrementando assim os contadores, já nas operações de desvio as portas presets são usadas para definir o valor do contador, porém antes que as portas porsets sejam escritas o contador e zerado, para garantir que a limpeza dos contadores ocorram antes da execução dos presets um bloco de atraso foi

adicionado, assim o pulso de clock alcança primeiro o reset depois o presets que compiam as entradas Pn para os contadores. A implementação completa deste circuito pode ser observada no Anexo II deste trabalho. Este é o único registrador que não segue a implementação do registrador demostrado anteriormente.

Acumulador (AC)

Por possuir somente um registrador de próposito geral este registrador é nomeado de acumulador, pois seu papel resume em armazenar o resultado das operações da ALU. Este registrador armazena 8 bits de dados, e é o operando das instruçãos executadas no processador, seu valor é modificado quando da execução das instruções. Sua implementação se resume a um conjunto de Flip-Flops do Tipo D com a porta de escrita controla da pela UC, seguindo o mesmo principio do registrador mencionado anteriormente.

Registrador de Dados da Memória (RDM)

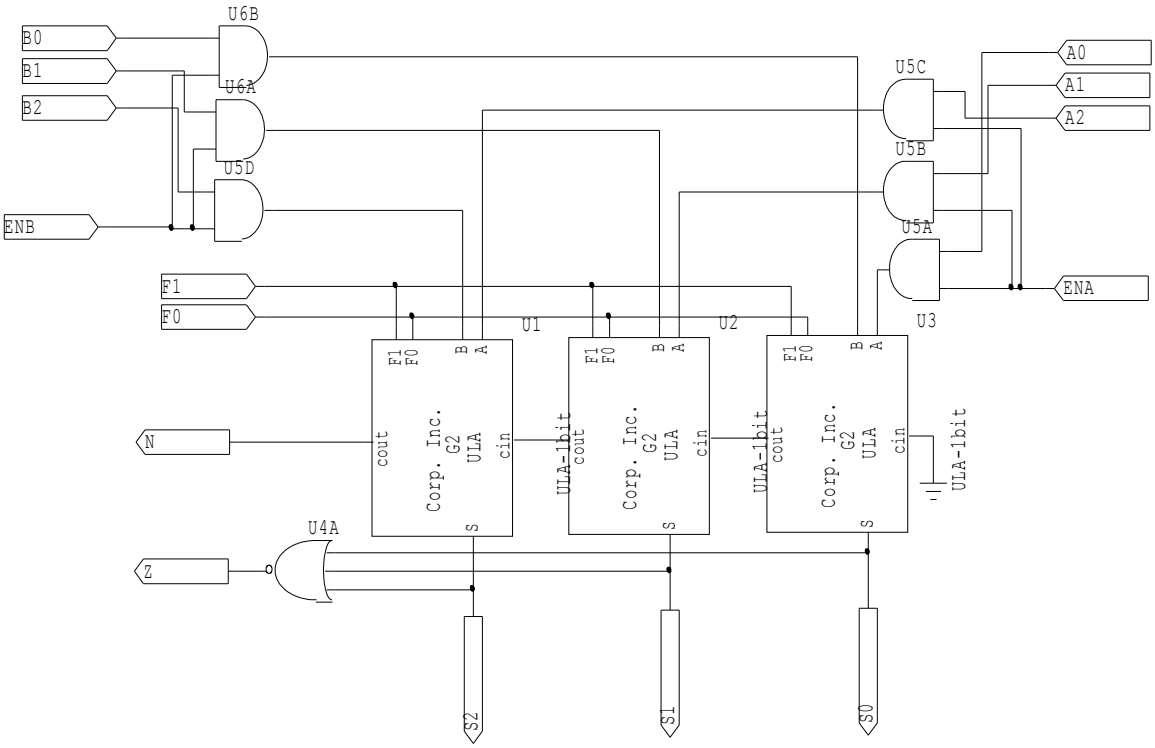
Este registrador é ligado à memória do sistema, e garda o valor do endereço de memória endereçado por REM ou por PC durante a busca. O tamanho deste registrador se limita a 8 bits, sendo este o maior bloco que se pode ler da memória do sistema a cada ciclo de execução. Sua implementação segue o mesmo modelo dos registradores de uso geral.

Registrador de Endereço de Memória (REM)

Este é o registrador de endereço de memória, para endereçar os 1024B de memória este registrador possui dez bits de dados (2^{10}), que endereçam blocos de 8 bits de memória que serão copiados para o RDM. Este registrador possui uma ligação com o PC para que nas operações de desvio seu valor seja inserido diretamente em PC, quando da ocorrência do desvio se conmfirmar (JN, JZ) ou se este for incondicional (JMP). Vale obervar que por possui dez bits de dados o valor deste processador é definido como os dois últimos bits do primero byte lido, combinado com os segundo byte que representa a instrução.

A Unidade Aritmética Lógica (ALU)

A ULA do sistema foi-se implementada através da associação de 8 ULAs de 1bit cada, que implementam as funções básicas de SOMA, OR, AND e NOT binários. Além disto a ULA adiciona algumas portas para habilitação/desabilitação das entradas A e/ou B, assim como sinais de controles que definem qual operação será executada, internamente estas entradas são utilizadas para controlar as ULAs de 1 bit, sendo os sinais de controle repassados para definir a operação a ser executada. A saída de cada ULA é ligada à saída da ULA de 8 bits respeitando a correspondência entre os bits tanto nas entradas quanto na saída das ULAs de 1 bit, além do resultado da operação a ULA retorna duas flags estado Z e N que indicam respectivamente se um resultado da operação é Zero ou se o valor retornado é um número negativo. Admitindo a possibilitade de expansão foram adicionadas as entrada Carry In e Carry Out na ULA, para que esta possa ser associada a outra ULA configurando assim uma ULA de 16 bits ou mais.



Na tabela ao lado pode-se conferir o código de cada operação implementada na ULA. Vale ressaltar que a operação NOT é unária e atua sobre a entrada A, sendo todas as demais operações binárias. A figura acima explicita uma versão reduzida da ULA com capacidade de efetuar cálculos de 3 bits, a ULA implementada aqui se resume tão somente á expansão do modelo acima para 8 bits.

Instrução	*F0*	*F1*
AND	**0**	**0**
NOT	**1**	**0**
OR	**0**	**1**
ADD	**1**	**1**

Tabela 1Código das operações da ALU

Implementação das ULA de 1 bit

A ULA de um bit é implemetada pela associação de uma somador completo, e circuitos de OR, NOT e AND que são acionados de acordo com o código de operação definidos por F0 e F1, cada uma das quatro combinações possivéis aciona uma das portas AND que filtram o resultado de cada operação de acordo com a tabela apresentada acima. Para que diversas ULAs possa ser associadas cada uma delas possuem um Carry-In e um Carry-Out através destas portas é possivél se combinar um número arbitrário destes elemento para formar uma ULA com capacidade para manipular números maiores.

Soma

A operação de soma é implementada internamente na ALU, através de um conjunto de somadores completos, a saída do Carrie-Out do bit mais significativo é utilizado para indicar se o número resultante é positivo ou negativo.

OR

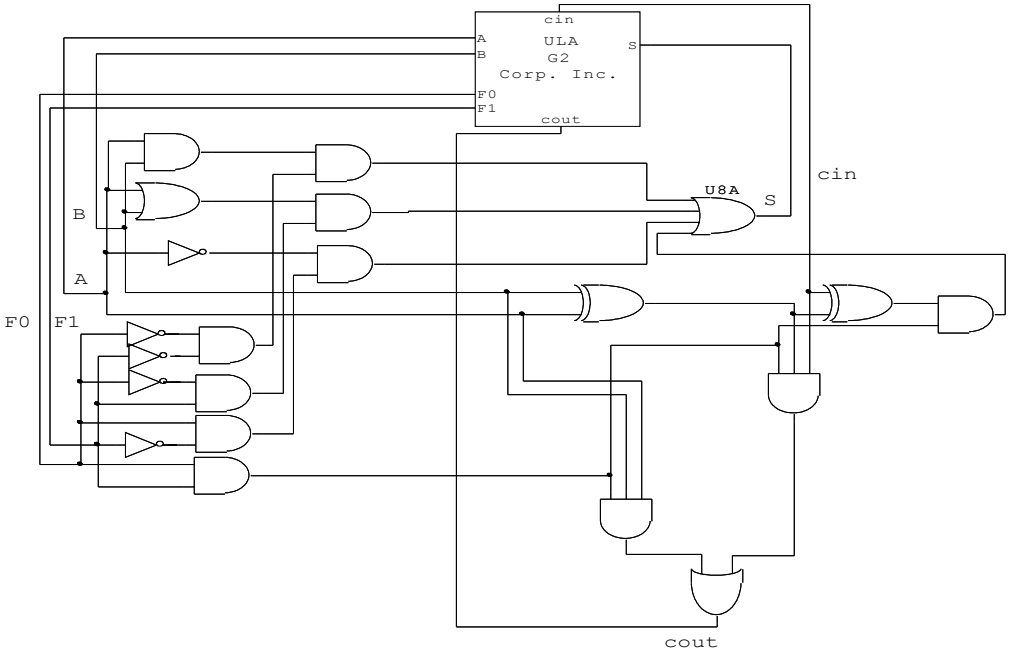
Esta operação é de fácil implementação, pois possui componentes digitais que as implementam diretamente.

AND

A função AND é realizada utilizando-se a porta lógica básica que a implementa.

NOT

Neste projeto a operação NOT se resume á inversão dos valores dos bits da entrada A da ALU, assim sua implementação é direta.



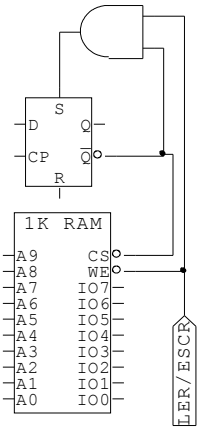
Flags de resultado (Z e N)

Cada operação executada na ALU além do seu resultado definem o resultado de dois flags neste projeto: N e Z, que indicam se o resultado da última operação foi um número negativo (acionando N), ou se foi o número zero (acionando Z). Estes resultados são usados como condicionais nas intruções de desvio JZ e JN e são copiados para UC para que sejam preservados entre as operações.

Caminho de Dados (Data Path)

Os componentes do sistema precisam de vias especiais para transmissão dos sinais, estas vias são constituidas de conjuntos de fios que interconectam os dispositivos, no diagrama apresentado no inicio deste trabalho estas vias são representadas pelas setas que ligam os dispositivos.

A Memória RAM



A memória do sistema possui capacidade para 1024 bytes, este espaço é compartilhado entre as instruções e os dados, sua implementação foge ao escopo deste trabalho sendo resumido aqui somente os sinais de controle utilizados para sua manipulação.

Há neste módulo 10 vias para definir o endereço que está sendo acessado (A0-A9), na face oposta se encontram as oito vias que são utilizadas para leitura/escrita do/no byte endereçado (IO0-IO7), a operação a ser executada é definida através de duas outras portas (CS e WE), a primeira entrada CS deve ser mantida em zero para que as operações de leitura escrita possam ser executadas, a segunda entrada WE alterna entre leitura (1) e escrita (0). Devido a um certo atraso no estabelecimento do sinal RWM (explicado posteriormente), o arranjo ao lado foi adicionado para que CS fique e 1 até que o sina de RWM fique estavél isto evita que a primeira instrução seja apagada durante a incialização do sistema.

Unidade de Controle (UC)

A Unidade de Controle é a parte do processador responsavél por gerar as microinstruções de cada operação implementada, é da responsabilidade deste circuito sequênciar as microinstruções nescessárias á executação da instrução. A execução de cada operação segue um ciclo definido como Busca, Decodificação e Execução.

Busca

O processo de busca consiste na recuperação da próxima instrução a ser executada na memória, a partir destes dados a Unidade de Controle define o que fazer dos dados. Devido á uniformidade das funções o processo de busca é um só para todas as operações. Isto levou a certa anomalias como a leitura de um operando “fantasma” para as funções que não possuem um, em contrapartida as microinstruções ficaram bastante reduzidas.

Decodificação

O processo de decodificação consiste na verificação do código de operação para determinar qual instrução está sendo executada, assim como o significado do seu operando, se este será utilizado para ler/escrever na memória, ou se deve ser silenciosamente descartado o operando da operação corrente, como em uma instrução NOP ou HLT. Neste projeto o código de operação é copiado para o RI durante o processo de decodificação. Outro ponto que vale ressaltar é que os bits mais significativos do 1º byte da operação são copiados para dois Flip-Flops tipo D, para que no próximo ciclo seja estes dois bits sejam combinados com o segundo byte da operação para que o endereço do operando seja recomposto para os seus 10 bits originais, isso faz com que o endereço de 10 bits do operando seja lido em duas partes, permitindo assim utilizar os dez bits de endereçamentos nescessários para acessar os 1024B de RAM do sistema.

Execução

Depois que os dados nescessários à operação estão carregados o próximo passo consiste na execução própriamente dita da instrução espécificada, esta operação é feita através da execução ordenada das microinstruções definidas para a operação corrente. Esta sequência de microinstruções foi-se de tal forma reduzida que optei por utilizar circuitos combinacionais para gerar a sequência de microoperações a serem executadas a partir do valor do OPCode da instrução corrente.

Micro-Operações

Micro operações são as operações funcionais ou atômicas de um processador, para execução de cada operação se é executado um conjunto de micro-operações previamente definidas pela UC.

Micro-instruções

Micro-instruções são os sinais de controle gerados pela UC para acionar os diversos componentes do sistema, permitindo assim a implementação de cada instrução, a geração destas micro-instruções podem ser feitas tanto através da leitura de dados gravados em ROM, quanto através de circuitos combinacionais, a implementação proposta para este trabalho usa um misto entre memórais ROM e circuitos combinacionais, para gerar tais sinais (12 externos à UC, e 1 de controle de estado da UC). Para facilitar a discussão da implementação da UC e suas micro-instruções adotou-se uma nomeclatura para permitir identificar cada um dos 13 sinais utilizados. Os sinais de controle gerados pela UC neste projeto foram nomeados como segue abaixo:

PC – Indica o acionamento da porta CLK do contador de programa, o que implica o incremento do valor deste registrador;

REM – Representa a porta de escrita do registrador de mesmo nome, que é acionada na transição positiva do clock;

RDM – Representa a porta de escrita do registrador de mesmo nome, que é acionada na transição positiva do clock;

RI – Está associada á porta de escrita de mesmo nome, bem como, á porta de escrita dos dois Flip-Flops que irão guardar os dois últimos bites do primeiro byte da instrução;

JMP – Este sinal foi ligado á porta JMP do PC, seu acionamento redefine o valor do registrador PC, para o valor das entradas deste registrador;

MRW – Alterna entre o modo de escrita e leitura da memória do sistema, simultâneamente ele desabilita ou reabilita a ligação entre o registrador AC e a memória;

MXE – Utilizado para controlar o MUX que alterna a entrada do endereço de memória entre o valor dos registradores PC e REM;

AC – Ativa a porta de escrita do registrador AC;

AEN – Ativa a entrada A da ULA, que esta ligada a registrador AC;

BEN – Ativa a entrada B da ULA, que esta ligada ao registrador RDM;

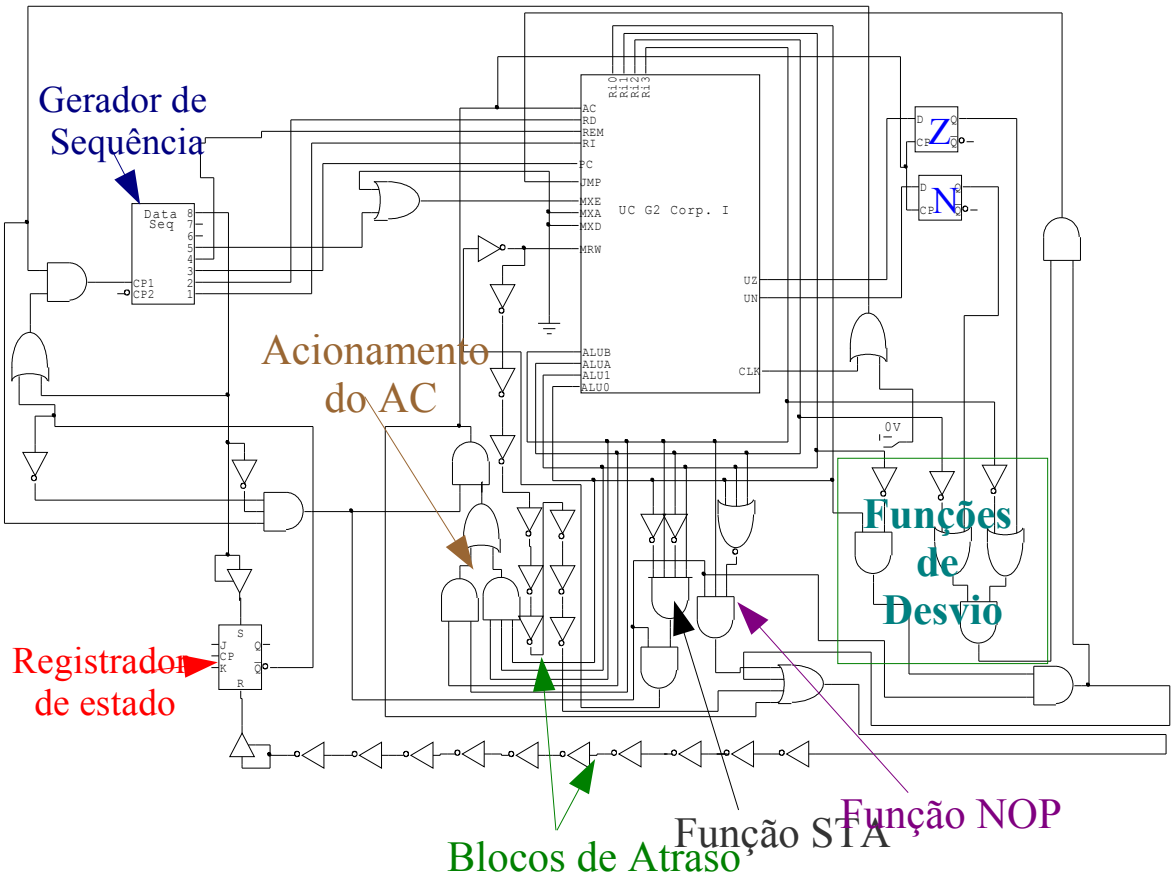
F0 e F1 – Define a função a ser executada pela ULA segundo a tabela apresentada;

FIM – Comando interno da UC que define o fim de cada estado, utilizado para alternar entre a busca e execução de cada instrução.

Implementação

A Unidade de Controle é implementada como uma máquina de estado, onde há dois estados possivéis: Busca e Decodificação representa o primeiro estado, e, Execução representa o segundo. Cada estado deve executar sua tarefa e então alterar o estado da máquina sinalizando que sua execução terminou através de um pulso de clock aplicado ao sinal de FIM, no caso da busca este ciclo é único, porém para cada operação a Execução pode levar a diferentes comportamento de acordo com o OPCODE vigente.

A figura abaixo identifica os principais blocos do sistema. Os blocos de atraso são nescessários para impedir que a transição de estado ocorra antes que o sinal de controle se estabilize fora da UC. Vale esclarecer que o sinal de clock (CLK) é alternado entre o bloco de decodificação e o bloco de execução sendo assim o sinal de CLK mostrado nas tabelas de micro-operações so estão ativos para um dos blocos.



Lista das Micro-instruções

As tabelas abaixo exprimem as micro-instruções geradas durante a sequência de busca e execução das operações, A primeria tabela detalha o processo de busca, sendo que todas as operações deste processador seguem este roteiro idependente de ter ou não um segundo operando.

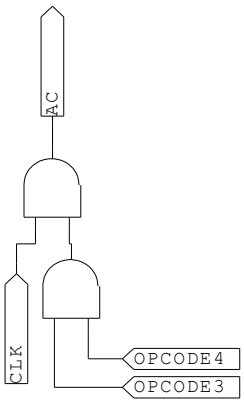
#	1	2	3	4	5	6	7	8	9
PC	0	1	0	0	0	0	1	0	0
REM	0	0	0	0	0	1	0	0	0
RDM	1	0	0	1	0	0	0	1	0
RI	0	1	0	0	0	0	0	0	0
JMP	0	0	0	0	0	0	0	0	0
MRW	1	1	1	1	1	1	1	1	1
MXE	0	0	0	0	1	1	1	1	1
AEN	0	0	0	0	0	0	0	0	0
AC	0	0	0	0	0	0	0	0	0
BEN	0	0	0	0	0	0	0	0	0
F0	0	0	0	0	0	0	0	0	0
F1	0	0	0	0	0	0	0	0	0
FIM	0	0	0	0	0	0	0	1	0

Os campos em cinza indicam que estes valores não são alterados durante a sequência de busca e decodificação.

Abaixo podemos observar a micro-instrução gerada por cada operação, observe que as micro-instruções da ULA correspondem ao código de operação vigente, observe também que os valores definidos no final do processo de busca e decodificação continuam constantes durante o processo de execução, e só serão alterados quando o ciclo de busca recomeçar, a função HLT é a única que não aciona o FIM pois ela prossegue indefinidamente, impedindo que a próxima intrução possa ser executada.

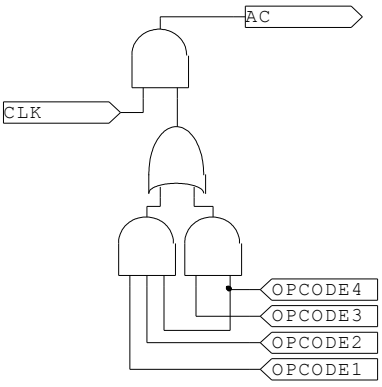
	ADD	OR	AND	NOT	LDA	STA	JMP	JN	JZ	NOP	HLT
PC	0	0	0	0	0	0	0	0	0	0	0
REM	0	0	0	0	0	0	0	0	0	0	0
RDM	0	0	0	0	0	0	0	0	0	0	0
RI	0	0	0	0	0	0	0	0	0	0	0
JMP	0	0	0	0	0	0	CLK	CLK ^ N	CLK ^ Z	0	0
MRW	1	1	1	1	1	CLK	1	1	1	1	1
MXE	1	1	1	1	1	1	1	1	1	1	1
BEN	1	1	1	1	1	1	0	0	1	1	0
AEN	1	1	1	1	0	0	0	1	0	0	0
F1	1	1	0	0	1	1	0	0	0	0	0
F0	1	0	0	1	1	0	1	1	1	0	0
AC	CLK	CLK	CLK	CLK	CLK	0	0	0	0	0	0
FIM	CLK	CLK	CLK	CLK	CLK	CLK	CLK	CLK	CLK	CLK	0

Funções matemáticas



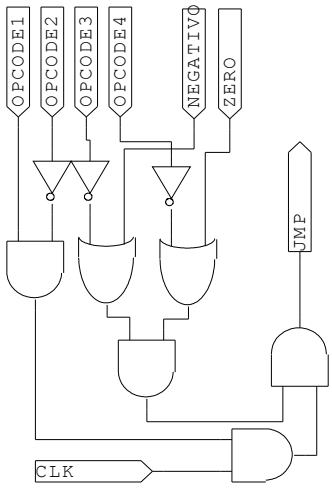
Todas as funções que operam sobre o valor de AC diferem somente no código de operação da ALU, como este código é gerado através do OPCODE vigente, o que resta é definir a condição para efetivação do pulso de clock que aciona a escrita em AC, assim observou-se a correspondência dos dois bits mais significativos do OPCODE, que estão ligados para todas as operações matemáticas, assim adicionou-se uma porta AND com as entrada alimentadas pelos dois últimos bits do OPCode, e a sua saída foi associada à entrada de outra porta AND que tem como segunda entrada o Clock do sistema como ilustra a figura ao lado. Somente com este circuito definiu-se as microperações de todas as operações matemáticas.

Funções Load e Store



A carga de uma valor da memória em AC é muito implementada através de um artificil bem simples, o OPCODE de LDA foi definido para que corresponde-se á operação de soma na ULA com a entrada B ativa e a entrada A desativada (1011), assim o valor de AC seria definido como B + 0, ou seja o valor de B. Para implementar tal comportamento bastou adicionar mais uma condição ao circuito da operações matemáticas que resultou no circuito ao lado.

Já a função STA tem um comportamento muito peculiar, sua função consiste na inversão do valor de MRW, quando isto a memória do sitema passa para o modo de escrita e AC é ligado à sua entrada através de 3-States que se ativam quando a inversão ocorre, para que este sinal permaneca por um periódo maior são adicionados blocos de atrasos, que nada mais são que um conjunto de portas AND, esta implementação pode ser visualizada na ilustração da UC apresentada anteriormente.

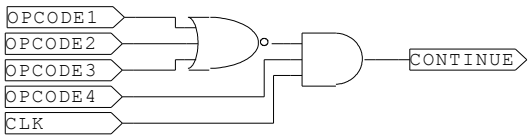


Funções de Desvio

Aqui outra coincidência proposital de valores permitiu que todas as funções de desvio fossem tratadas de forma única, por possuírem o OPCODE no formato XX10 elas foram implementadas como mostra a figura ao lado, observe que a condição para que o desvio ocorre é que o 3º bit esteja zerado ou a flag N esteja ativa e o 4º bit seja zero ou a flag Z esteja ativa, assim o desvio incondicional (JMP) tem os dois últimos bits zerados (0001), o desvio se negativo tem o 3º bit em 1 e o 4º em Zero (0101), e o desvio se zero tem o 3º bit com zero e o 4º bit com o valor 1 (1001). Vale ressaltar que a designação de 3º e 4º bit se referem de forma inversa à sua posição no número assim o bit que representa 2^0 é o primero e 2^3 o quarto bit por isso aparecem no inicio do número apresentado.

Funções do Sistema

Há duas função especias no conjunto proposto, NOP e HLT, a primeira serve para adicionar atrasos ao ciclo de execução, não tendo nenhum efeito colateral, a segunda suspende a execução do sistema. Para implementar a função NOP basta verificar se o OPCODE vigente é da função NOP então mudar o estado da UC para Busca e Decodificação como todas as outras funções fazem após a execução. Observe que o sinal de continue foi suprimido nas funções acima mais eles podem ser observados na figura que descreve a UC de forma completa.



A função HLT é a de mais simples implementação, ..., ela não tem uma implementação, assim como a UC é implementada como uma máquina de estados ela vai ficar no estado de execução sem fazer nada indefinidamente, pois não há nenhuma condição que a faça voltar para o estado de busca, o código de HLT foi definido como 0000, assim sempre que não houver nenhuma função definida é suspendida a execução do programa.

Conclusão

Num projeto de processador tanto o caminho de dados e os elementos de estados e combinacionais possuem uma relevância muito grande, a correta disposição destes componentes possibilitam uma redução siginificativa no tamanho e complexidade do projeto, porém deve-se medir o impacto que cada escolha fará ao projeto no tocante à velocidade, e custo de implementação. Apesar de ser um projeto reduzido de processador, como não poderia deixar de ser enquanto projeto didático, sua implementação possibilitou um profundo entendimento sobre como funciona um processador convencional.

Para tornar a implementação mais enxuta algumas concessões foram feitas, como por exemplo a uniformidade no tamanho das funções, o que ocasiona uma perda de espaço na memória já que algumas operações poderiam ocupar somente um byte, e também reduzi o tempo de busca pela metade em operações como NOT, NOP e HLT.

Referências Bibliográficas

IRVINE, Kip R., *Assembly Language for Intel-Based Computers,* Fourth Edition, Prentice Hall, 2003.

PATTERSON, David A. e HENNESSY, John L., *Organização e projeto de computadores a interface hardware/software*, Segunda edição. Rio de Janeiro: LTC, 2000.

GAJSKI, Daniel D., *Principles Of Digital Design*, Prentice-Hall Inc., 1997

STALLINGS, Willian, *Computer Organization and Architecture Design for Performance*, Sixth Edition, Prentice Hall, 2003.

TANENBAUM, Andrew. S., *Organização Estruturada de Computadores*, Quarta Edição. Rio de Janeiro: Prentice-Hall do Brasil, 1999.

IDOETA, Ivan V., CAPUANO, Francisco F., *Elementos de Eletrônica Digital*. 6ª Edição. São Paulo: Érica, 1984.

TOCCI, R. J. *Sistemas Digitais: Princípios e Aplicações* Ed.LTC, 2000, 7° edição

WEBER, Raul Fernando, *Fundamentos de Arquitetura de Computadores*, 2ª edição. Porto Alegre: Editora Sagra Luzzatto, 2001.

Matériais Técnicos
Intel Architecture Software Developer's Manual Volume 1: Basic Architecture
Intel Architecture Software Developer's Manual Volume 2: Instruction Set Reference
IA-32 Intel® Architecture Optimization Reference Manual

Anexo I

Programa de Teste:

Abaixo segue uma programa usado para testar todas as instruções do sistema:

Endereço	*Valor Inicial*		
	Binário	*Hexa*	*Dec*
600	00110010	32	50
601	01101011	6B	107
602	00010101	15	21
603	00100101	25	37
604	11000101	C5	197
605	01000011	43	67
606	01111000	78	120
607	01011010	5A	90
610	00000000	00	0

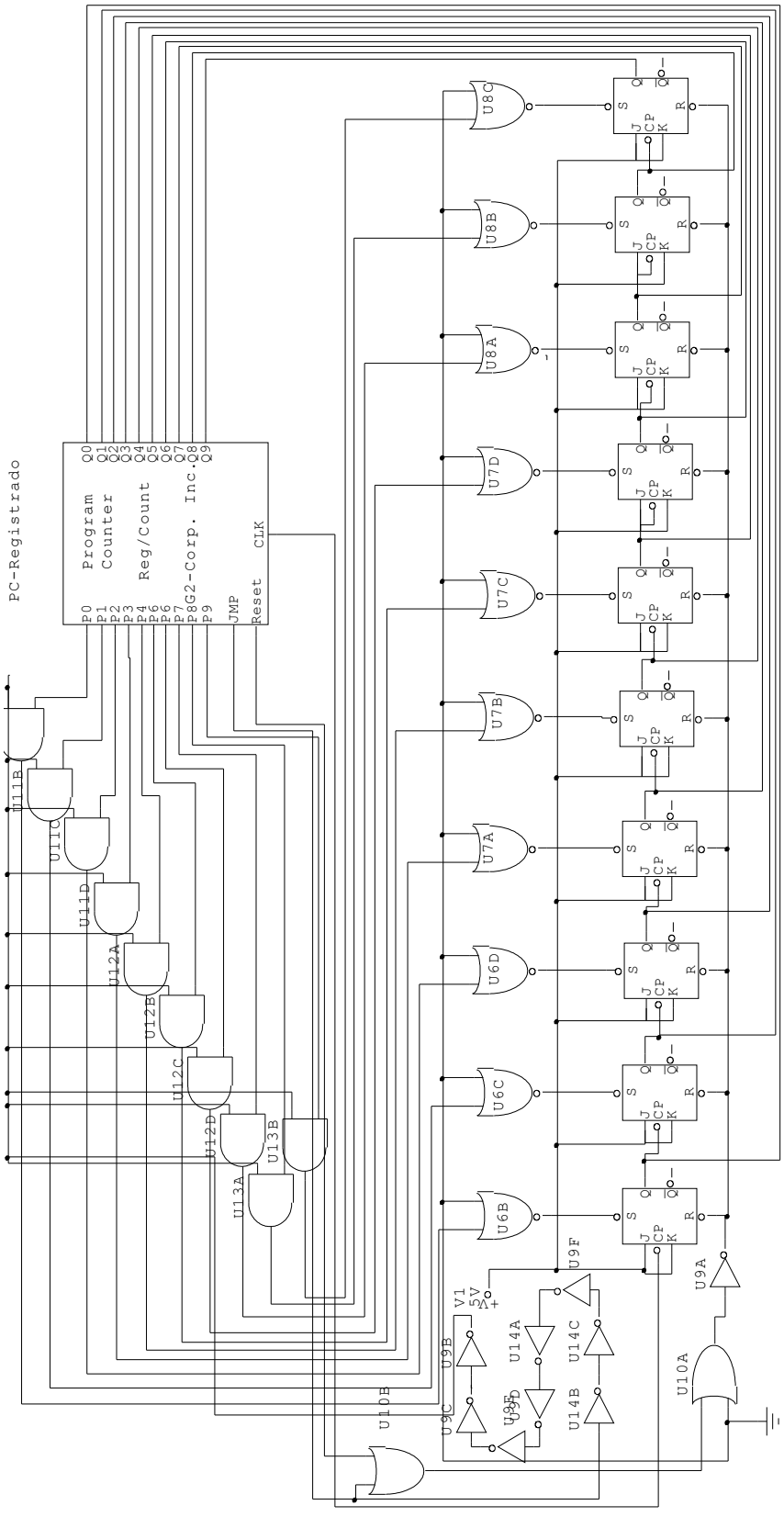
Endereço	*Valor Final*		
	Binário	*Hexa*	*Dec*
700	-	-	-
701	11001101	CD	205
702	11111111	FF	255
703	00000000	00	0
704	00010101	15	21

Endereço	*Operação*
0	**NOP;**
2	**LDA 600;**
4	**NOT;**
6	**STA 701;**
8	**OR 600;**
10	**STA 702;**
12	**LDA 602;**
14	**ADD 603;**
18	**AND 604;**
20	**JZ 30;**
22	**HLT;**
24	**AND 610;**
26	**STA 700;**

Endereço	*Operação*
28	**HLT;**
30	**STA 703;**
32	**LDA 605;**
34	**ADD 606;**
36	**JN 22;**
38	**ADD 607;**
40	**JN 44;**
42	**HLT;**
44	**STA 704;**
46	**NOP;**
48	**NOP;**
50	**HLT;**
52	**-**

Anexo II

Modelo de Implementação do Contador de Programa:


PC-Registrado
Program Counter
Reg/Count
P8G2-Corp. Inc.
P0 P1 P2 P3 P4 P5 P6 P7 P8 P9 JMP Reset CLK
Q0 Q1 Q2 Q3 Q4 Q5 Q6 Q7 Q8 Q9
U11B
U11C
U11D
U12A
U12B
U12C
U12D
U13A
U13B
U10B
U8C
U8B
U8A
U7D
U7C
U7B
U7A
U6D
U6C
U6B
U9F
U9A
U10A
U9B
U9C
U9D
U9E
U14A
U14B
U14C
V1
5V
S
J
CP
K
R
Q