Relatório Relógio

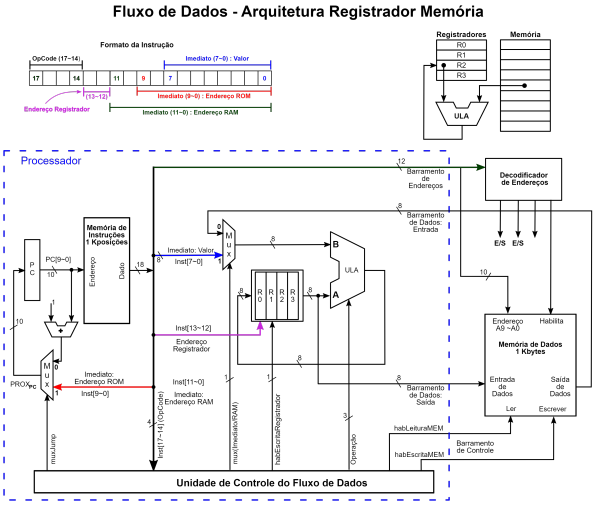
Arquitetura do processador

A entrega intermediária do projeto consiste na implementação de um processador feito na arquitetura Registrador-Memória. Nessa primeira entrega, o objetivo que tivemos era de realizar um contador, em que, ao apertar o botão, ele carrega mais um no display 7-seg.

Como utilizamos a arquitetura de registrador memória, obtivemos algumas vantagens em comparação à arquitetura de acumulador, já que tivemos uma redução no número de instruções utilizadas e também nos ajudará na projeção do relógio.

A entrega intermediária foi realizada com os seguintes adicionais:

- Processador com a arquitetura Registrador-Memória
- Implementação da instruções extras: JLT, CLT e ADDI
- Montador assembly
- Aceleração do tempo



Manual de Botões

Key 0 → Acerela a frequência do tempo

Key 3 → Zera o relógio se em todas as casas

Total de instruções e sua sintaxe

Instrução	Mnemônico	Código Binário
Sem Operação	NOP	000
Carrega valor da memória para A	LDA	0001
Soma A e B da memória para A	SOMA	0010

Subtrai B de A	SUB	0011
Carrega valor imediato para A	LDI	0100
Salva valor de A para memória	STA	0101
Desvio de execução	JMP	0110
Desvio condicional de execução	JEQ	0111
Comparação	CEQ	1000
Chamada de Sub Rotina	JSR	1001
Retorno de Sub Rotina	RET	1010
Mascara bits e salva valor no imediato	ANDI	1011
Compara se menor que	CLT	1100
Desvia se menor que	JLT	1101
Doma do acumulador com imediato	ADDI	1110

Formato das instruções (distribuição dos campos na palavra de instrução)

OpCode	**Registrador**	**Imediato**
4 bits	2 bits	8 bits
(14 ~ 11)	(10 ~ 9)	(7 ~ 0)

Fluxo de dados para o processador, com uma explicação resumida do seu funcionamento

Com base na nossa arquitetura de registrador memória, o fluxo de dados se começa pela busca de uma instrução no Program Counter, nele armazenamos temporariamente os dados, endereços e instruções. Lembrando que as instruções se realizam por uma ordem, ao menos que venha uma instrução do tipo Jump, fazendo com que a sequência mude.

Depois é realizada a decodificação dessas instruções, habilitando os componentes necessários para realizar o que foi pedido. Assim o processador lê e realiza as instruções por meio da ULA, MUX e o banco de registradores, lembrando que ao aumentar o número de registradores, otimizamos nosso processador de modo que uma tarefa acaba utilizando menos clocks. Após isso o processador escreve o resultado e o ciclo se inicia novamente.

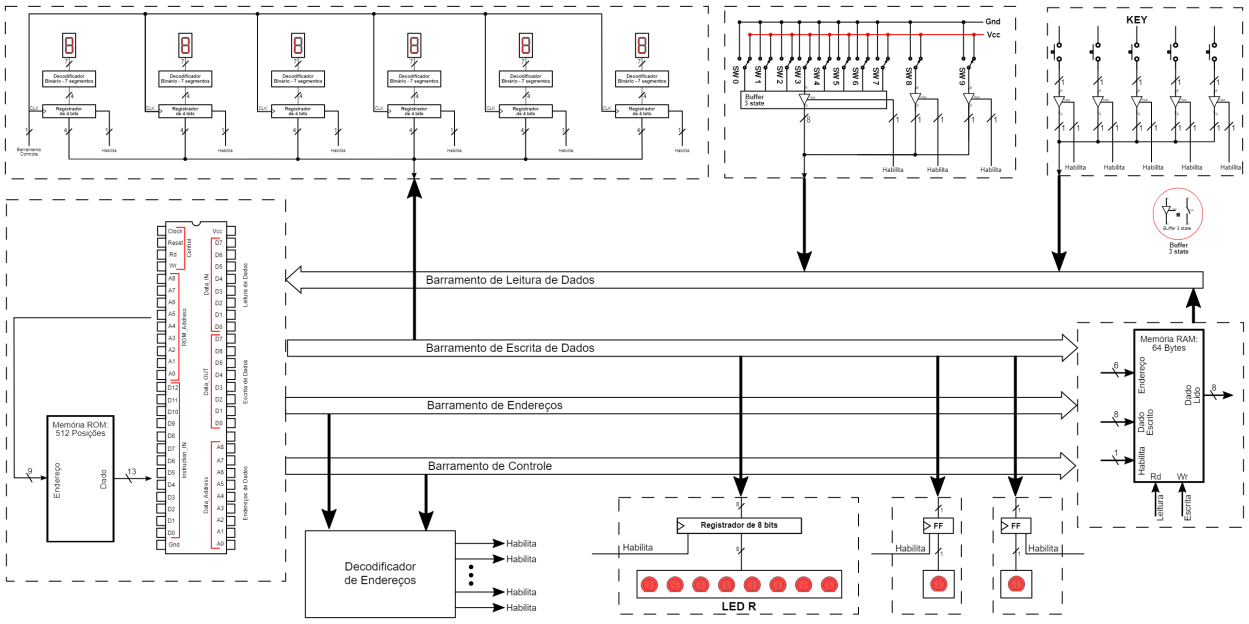
Listagem dos pontos de controle e sua utilização

Ponto de Controle	**Utilização**
habEscritaMEM	Habilita escrita da memória RAM
habLeituraMEM	Habilita leitura da memória RAM
Operação	Definir qual operação será realizada na ULA: Soma (00), Sub (01) ou Passa (10)
habEscritaRegistrador	Habilita a escrita nos registradores
Mux(Imediato/RAM)	Definir a entrada B da ULA (Imediato ou memória RAM)
MuxJump	Seleção de Incremento do PC (00), Instrução (01) e Endereço de Retorno (10)

	Hab Ret	JMP	RET	JSR	JEQ	JLT	SelMux
NOP	0	0	0	0	0	0	X
LDA	0	0	0	0	0	0	0
SOMA	0	0	0	0	0	0	0
SUB	0	0	0	0	0	0	0
LDI	0	0	0	0	0	0	1
STA	0	0	0	0	0	0	0

JMP	0	1	0	0	0	0	X
JEQ	0	0	0	0	1	0	X
CEQ	0	0	0	0	0	0	0
JSR	1	0	0	1	0	0	X
RET	0	0	1	0	0	0	X
ANDI	0	0	0	0	0	0	1
CLT	0	0	0	0	0	0	0
JLT	0	0	0	0	0	1	X
ADDI	0	0	0	0	0	0	1

Rascunho do diagrama de conexão do processador com os periféricos



Temos que as Keys e SWs estão conectados ao barramento de leitura de dados, já os Display 7seg e os LEDs estão cobectados ao barramente de escruta de dados. Por fim, o Decodificados de Endereços conecta os barramentos de Endereços e Controle.

Rascunho do mapa de memória

Mapa de Memória

Endereço em Decimal	Periférico	Largura dos Dados	Tipo de Acesso	Bloco (Página) de Memória
0 ~ 63	RAM	8 bits	Leitura/Escrita	0
64 ~ 127	Reservado	--	--	1
128 ~ 191	Reservado	--	--	2
192 ~ 255	Reservado	--	--	3
256	LEDR0 ~ LEDR7	8 bits	Escrita	4
257	LEDR8	1 bit	Escrita	4
258	LEDR9	1 bit	Escrita	4
259 ~ 287	Reservado	--	--	4
288	HEX0	4 bits	Escrita	4
289	HEX1	4 bits	Escrita	4
290	HEX2	4 bits	Escrita	4
291	HEX3	4 bits	Escrita	4
292	HEX4	4 bits	Escrita	4
293	HEX5	4 bits	Escrita	4
294 ~ 319	Reservado	--	--	4
320	SW0 ~ SW7	8 bits	Leitura	5
321	SW8	1 bit	Leitura	5
322	SW9	1 bit	Leitura	5
323 ~ 351	Reservado	--	--	5
352	KEY0	1 bit	Leitura	5
353	KEY1	1 bit	Leitura	5
354	KEY2	1 bit	Leitura	5
355	KEY3	1 bit	Leitura	5
356	FPGA_RESET	1 bit	Leitura	5
357 ~ 383	Reservado	--	--	5
384 ~ 447	Reservado	--	--	6
448 ~ 510	Reservado	--	--	7
510	Limpa Leitura KEY1	--	Escrita	7
511	Limpa Leitura KEY0	--	Escrita	7

Fonte do programa em assembly

O nosso código foi estruturado da seguinte maneira: setup, loop e subrotinas. Inicialmente declaramos as principais variáveis em endereços da memória específicos (8 a 15) e também alocamos outros endereços (0 a 6) que servirão para armazenarmos as nossas variáveis do contador (como os valores das unidades, dezenas, etc, assim como uma flag de overflow).

Um ponto importante de ressaltar é que, como fizemos a implementação da instrução ANDI (a qual compara bit a bit com base em uma máscara) acabamos tendo um ganho grande em eficiência na hora de transbordar quando o contador atingia o limite de determinada casa decimal. Isso ocorreu já que a própria instrução acaba realizando essa comparação e armazenando seu resultado para a verificação posterior, ou seja, se a zerassemos e adicionariamos uma unidade na próxima casa decimal.

```
LDI $0, R0
LDI $1, R1
LDI $6, R2
LDI $10, R3
STA @256, R0
STA @257, R0
STA @258, R0
STA @288, R0
STA @289, R0
STA @290, R0
STA @291, R0
STA @292, R0
STA @293, R0
STA @0, R0
STA @1, R0
STA @2, R0
```

```
STA @3, R0
STA @4, R0
STA @5, R0
STA @6, R0
STA @7, R0
STA @8, R1
STA @9, R3
STA @10, R2
STA @11, R2
STA @12, R2
STA @13, R2
STA @14, R2
STA @15, R2
STA @16, R2
LDI $2, R0
STA @17, R0
LDI $4, R0
STA @18, R0
LDI $24, R0
STA @19, R0

LOOP_PRINCIPAL:
NOP
LDA @352, R0
ANDI @1, R0
CEQ @7, R0
JEQ @LEITURA_KEY1
JSR @INCREMENTO
NOP

CHAMA_ATUALIZA_DISPLAY:
JSR @ATUALIZA_SEVEN_SEG
JMP @LOOP_PRINCIPAL

INC:
STA @511, R0
LDA @6, R1
CEQ @8, R1
JEQ @120

INC_UNIDADE:
LDA @0, R0
SOMA @8, R0
CEQ @9, R0
JEQ @INC_DEZENA
STA @0, R0
RET

INC_DEZENA:
LDA @7, R0
STA @0, R0
LDA @1, R0
SOMA @8, R0
CEQ @9, R0
JEQ @INC_CENTENA
STA @1, R0
RET

INC_CENTENA:
LDA @7, R0
STA @1, R0
LDA @2, R0
SOMA @8, R0
CEQ @9, R0
JEQ @INC_UNIMILHAR
STA @2, R0
RET

INC_UNIMILHAR:
LDA @7, R0
STA @2, R0
LDA @3, R0
SOMA @8, R0
CEQ @9, R0
JEQ @INC_DEZMILHAR
STA @3, R0
RET

INC_DEZMILHAR:
LDA @7, R0
STA @3, R0
```

```
LDA @4, R0
SOMA @8, R0
CEQ @9, R0
JEQ @INC_CENTMILHAR
STA @4, R0
RET
INC_CENTMILHAR:
LDA @7, R0
STA @4, R0
LDA @5, R0
SOMA @8, R0
CEQ @9, R0
JEQ @OVERFLOW
STA @5, R0
RET

CONFERE_24:
LDA @5, R1
CEQ @17, R1
JEQ @REINICIAR_CONTAGEM_24
CEQ @9, R0
JEQ @INC_CENTMILHAR
STA @4, R0
JMP @FIM_INCREMENTO

REINICIAR_CONTAGEM_24:
LDA @7, R0
STA @0, R0
STA @1, R0
STA @2, R0
STA @3, R0
STA @4, R0
STA @5, R0

FIM_INCREMENTO:
RET

ATUALIZA_SEVEN_SEG:
LDA @0, R0
LDA @1, R1
LDA @2, R2
STA @288, R0
STA @289, R1
STA @290, R2
LDA @3, R0
LDA @4, R1
LDA @5, R2
STA @291, R0
STA @292, R1
STA @293, R2
LDA @6, R0
CEQ @7, R0
JEQ @FIM_ATUALIZA_TELA
LDA @8, R2
STA @257, R2

FIM_ATUALIZA_TELA:
RET

REINICIAR_CONTAGEM:
LDA @7, R0
STA @0, R0
STA @1, R0
STA @2, R0
STA @3, R0
STA @4, R0
STA @5, R0
STA @6, R0
STA @257, R0
STA @258, R0
RET

LIM_UNIDADE:
LDA @8, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_UNIDADE
STA @10, R2
```

```
STA @510, R0

LIM_DEZENA:
LDI $2, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_DEZENA
STA @11, R2
STA @510, R0

LIM_CENTENA:
LDI $4, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_CENTENA
STA @12, R2
STA @510, R0

LIM_MILHAR:
LDI $8, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_UNIMILHAR
STA @13, R2
STA @510, R0

LIM_DEZMILHAR:
LDI $16, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_DEZMILHAR
STA @14, R2
STA @510, R0

LIM_CENTMILHAR:
LDI $32, R1
STA @256, R1
LDA @353, R0
ANDI @1, R0
CEQ @7, R0
LDA @320, R2
JEQ @LIM_CENTMILHAR
STA @15, R2
STA @510, R0
LDA @7, R3
STA @256, R3
RET

LESS_24_HRS:
LDA @14, R0
LDA @15, R1
LDA @7, R2

CONCAT_LIMITE:
ADDI $10, R3
SUBI $1, R1
CEQ @7, R1
JEQ @CHECK_LIM
JMP @CONCAT_LIMITE

CHECK_LIM:
SOMA @14, R3
CLT @19, R3
JLT @FIM_LIMITE_DESPERTA
CEQ @19, R3
JEQ @EH_24_HORAS_LIMITE
LDA @8, R0
STA @258, R0
JMP @LIM_UNIDADE
```

```
EH_24_HORAS_LIMITE:
LDA @8, R0
STA @258, R0
LDA @7, R0
STA @14, R0
STA @15, R0
LDA @10, R0
CEQ @7, R0
JEQ @CONFERE_SEG2_LIMITE
JMP @LIM_UNIDADE

COMPARA_MENOR_24:
LDA @4, R0
LDA @5, R1
LDA @7, R2

CONCAT:
ADDI $10, R3
SUBI $1, R1
CEQ @7, R1
JEQ @CHECK
JMP @CONCAT

CHECK:
SOMA @4, R3
CLT @19, R3
JLT @FIM_CONFIG_HORA
CEQ @19, R3
JEQ @EH_24_HORAS
LDA @8, R0
STA @258, R0
JMP @CONFIG_SEG1
```