

Insper  
Bacharelado em Engenharia da Computação

Adney Moura e Letícia Coêlho

Design de Computadores : Relatório Relógio

São Paulo  
Abril de 2023

## Sumário

1	Introdução	2
2	Funcionamento	3
3	Arquitetura	4
4	Mapa de Memoria	10
5	Conceitos atingidos no Projeto	13
6	Assembler	13
7	Link do vídeo	25

## 1 Introdução

Este relatório tem como objetivo apresentar um relógio construído em linguagem de montagem (assembly) e compilado em uma FPGA – 5CEBA4F23C7N. o relógio em questão foi implementado em assembly para maximizar a eficiência e desempenho do sistema, bem como permitir um maior controle sobre o hardware utilizado. A seguir, serão apresentados os detalhes do projeto, desde o desenvolvimento do código em assembly até a compilação e implementação na FPGA, bem como os resultados obtidos e as conclusões sobre o projeto.

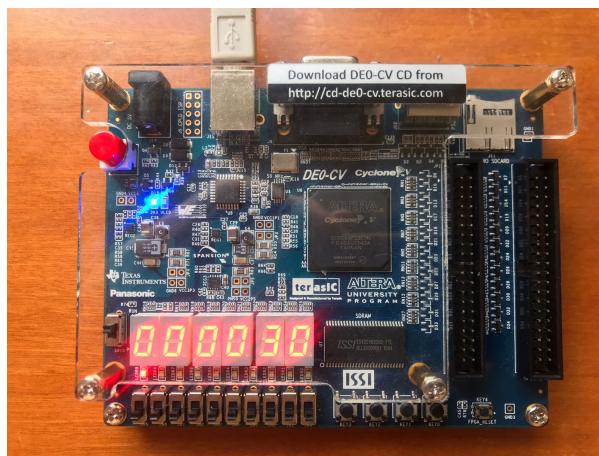


Figura 1: Imagem ilustrativa do funcionamento do relógio

## 2 Funcionamento

A construção da infraestrutura do microcontrolador fora implementada em VHDL e gravada na FPGA. O código é capaz de interagir diretamente com a leitura dos botões e chaves da FPGA, assim como a escrita nos LEDs e no Display. A figura 2 ilustra esses componentes.

O Display possui a funcionalidade de mostrar o indicativo da hora atual, assim como também, permitir que um novo horário seja configurado no relógio.

Dentre os botões, temos os seguintes usos:

1. KEY1 : Pressionar KEY1 possibilita a configuração do limite varrendo cada dígito do sistema. Ou seja , com a opção de "inserir horário atual" selecionada, pressionar essa botão confirma a mudança na casa decimal atual, e permite a modificação da próxima casa.
2. *FPGA RESET* : Reset do horário quando as 24hrs do relógio forem atingidas, reiniciando a contagem.

Utilizamos as chaves da FPGA para acelerar a base de tempo (intercalar a velocidade da contagem do relógio - SW9 ) e para dar o input dos valores da configuração "inserir hora atual", dígito por dígito. É realizado um filtro limitando o valor que o usuário pode setar na configuração do horário atual. Ou seja, o valor da unidade-segundo e unidade-minuto, podem ter o valor máximo igual a 9, dezena-segundo e dezena-minuto , podem ter valor no máximo igual a 5 e para o input das horas o valor de input deve está entre 0 e 24.

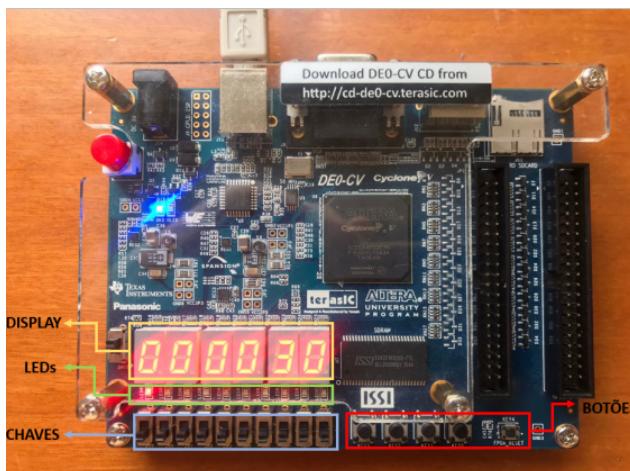


Figura 2: FPGA com indicação das funcionalidades utilizadas

### 3 Arquitetura

Nesse tópico será descrito como a arquitetura foi desenvolvida para que o relógio fosse construído.

O contador faz uso da arquitetura de processador baseada em registrador memória. Dessa forma ,uma operação ocorrem entre um registrador e uma posição de memória ou um valor imediato.O resultado dessa operação é armazenada no mesmo registrador utilizado como um dos argumentos.

O formato das instruções utilizadas pode ser visto na figura 3.



Figura 3: Formato das instruções utilizadas [1]

O formato de todas as possíveis operações realizadas pelo processador estão descritas na tabela abaixo.

Instrução	Uso	Simplificando	Descrição
NOP	NOP	-	Sem operação
LDA	LDA Rx @E	Rx = M[E]	Carrega valor da memória para Rx
SOMA	SOMA Rx @E	Rx = Rx + M[E]	Soma valor na memória com Rx
SUB	SUB Rx @E	Rx = Rx - M[E]	Subtrai Rx de valor memória
LDI	LDI Rx \$Valor	Rx = Valor	Carrega valor imediato para Rx
STA	STA @E Rx	M[E] = Rx	Salva valor de Rx na memória
JMP	JMP @E	-	Desvia execução para E
JEQ	JEQ @E	Igual = (R = M ? 1 : 0)	Desvio de execução condicional
CEQ	CEQ Rx @E	-	Compara Rx com um valor na memoria
JSR	JSR @E	-	Chamada de Sub Rotina
RET	RET	-	Retorno de Sub Rotina
OP <sub>AND</sub>	OP <sub>AND</sub> Rx @E	M[E] AND Rx	Operação AND entre memoria e Rx
CLT	CLT Rx @E	Menor = Rx < Mem[E] ? 1 : 0	Rx < MEM[E] , ativa Menor
JLT	JLT @E	Menor = 1 ? PC = E : Pc+=1	Menor =1 , salta para E
ADDI	ADDI Rx \$Valor	Rx = Valor	Carrega valor do imediato em Rx

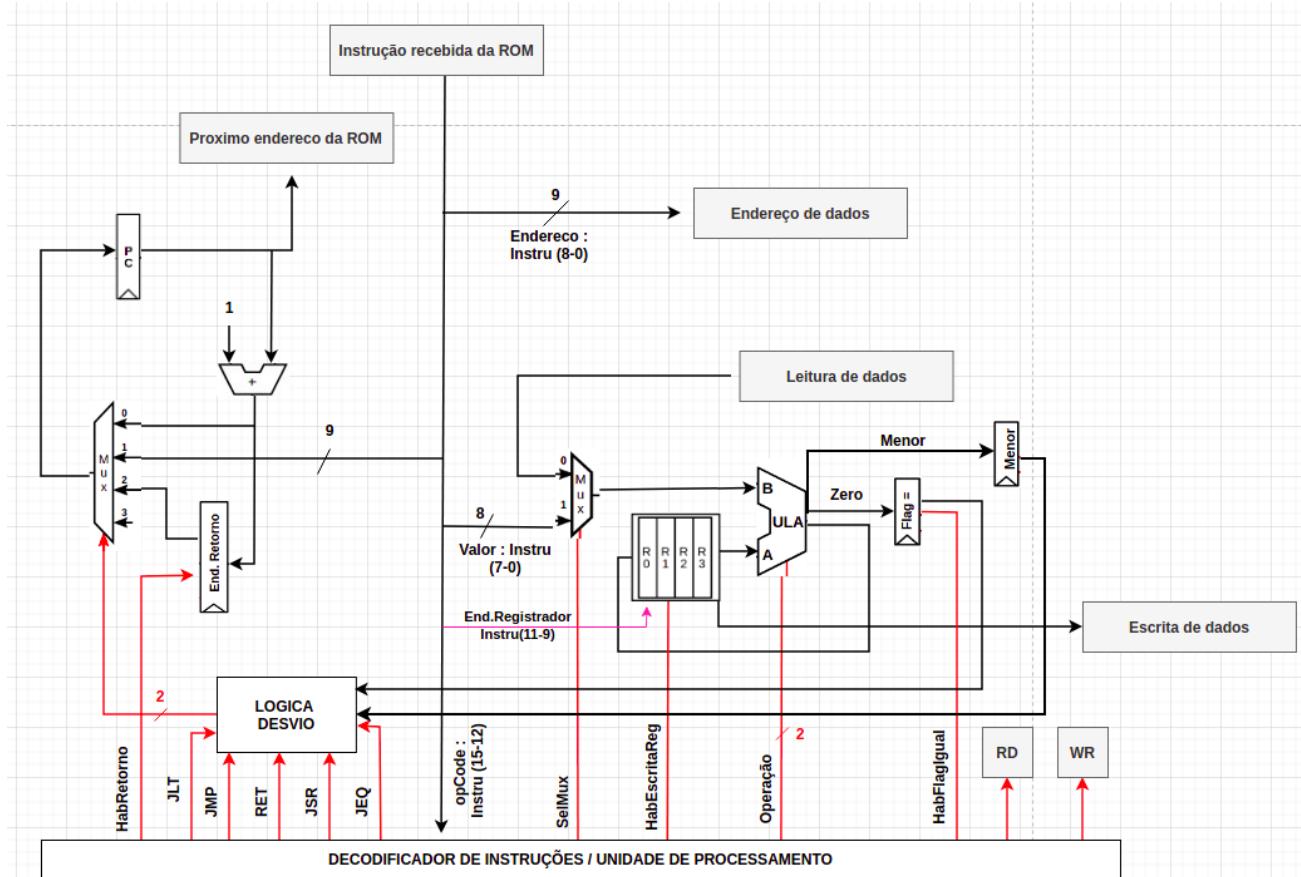


Figura 4: Arquitetura do Processador [Adaptada] [1]

Importante ressaltar, que apesar de possuirmos na imagem apenas 3 registradores, a nossa arquitetura possui um banco de registradores de tamanho 8. Com o processador bem definido, podemos mostrar a arquitetura "mais global", abstraindo o componente CPU descrito anteriormente, como pode ser demonstrado na figura 5.

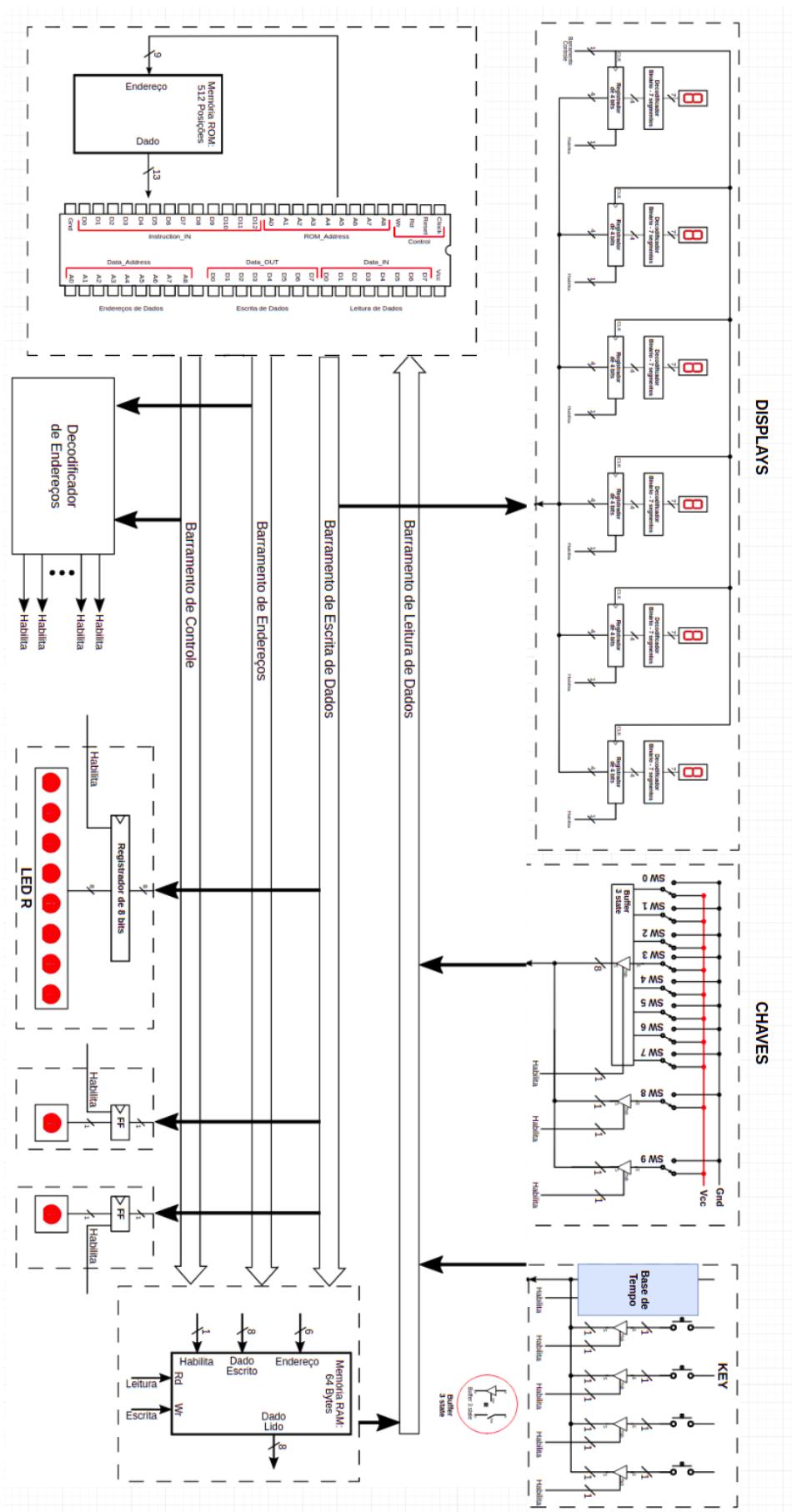


Figura 5: Arquitetura completa do Projeto [Adaptada] [1]

Apesar de não estar explícito na imagem, os botões utilizados no projeto possuem um circuito que possibilita a memorização (FlipFlop) dos valores e a retirada de ruídos durante o apertar do botão (edgeDetector). Esse circuito pode ser representado para todas as chaves como mostrado abaixo:

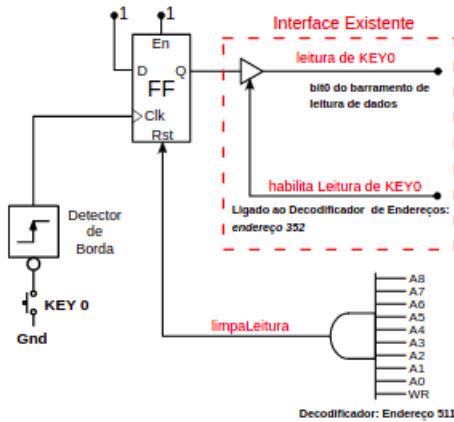


Figura 6: Circuito de memorização de debounce dos botões [1]

É possível notar na arquitetura completa mostrada anteriormente uma "caixa preta" denominada Base de Tempo. Em nosso projeto, o componente base de tempo é o responsável por "contar" os segundos de nosso relógio. Esse componente ocupa um endereço , que quando acessado, preenche o barramento de dados com a informação de haver passado 1 segundo (1) ou não (0). O endereço escolhido para tal componente foi o mesmo endereço de acesso da chave KEY0, a limpeza de seu flipflop também é utilizado para limpar o flipflop da base de tempo.

**Componente base de Tempo**

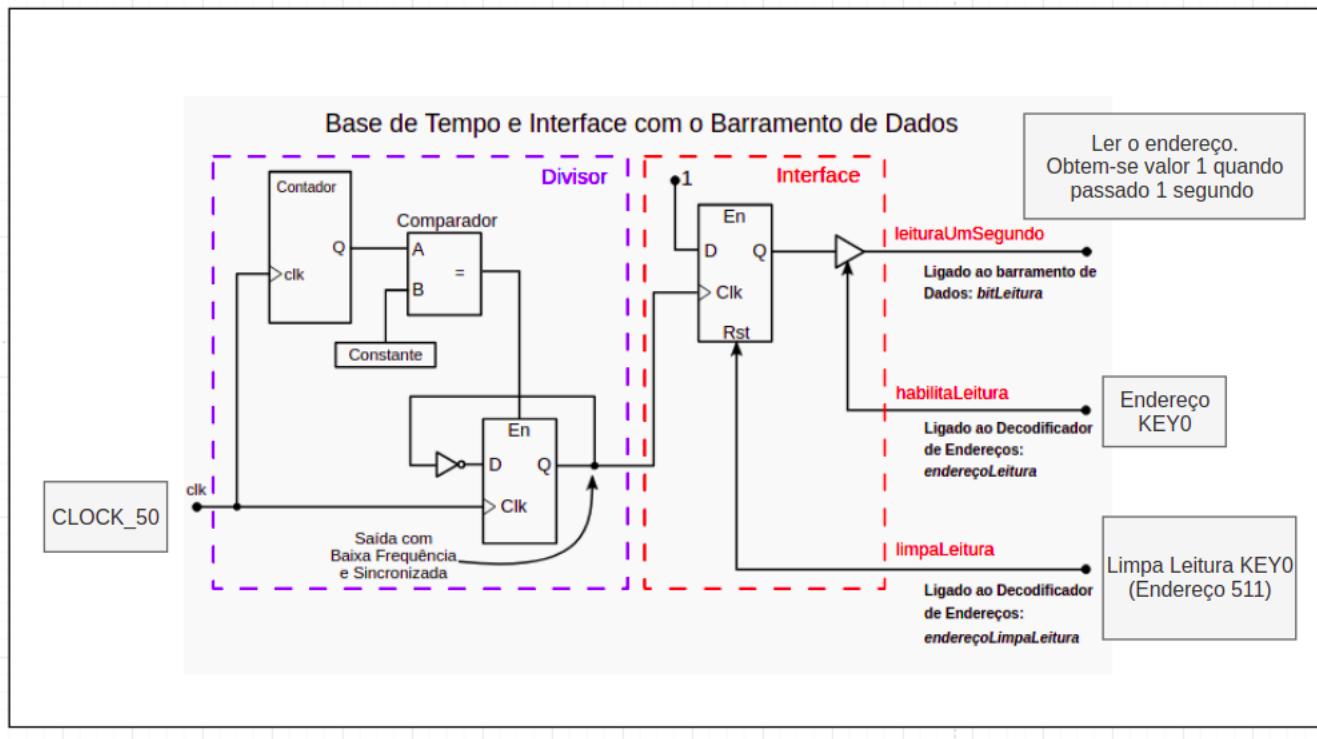


Figura 7: Componente Base de Tempo [1]

Como descrito no projeto anteriormente, possuímos duas forma de contagem em nosso relógio, a comum (segundo a segundo) e uma contagem mais rápida (para verificar o comportamento do relógio de forma rápida e eficaz). Para conseguirmos ter o acesso a essas duas bases de tempo, o seguinte diagrama foi construído.

### Lógica para intercalar base de tempo

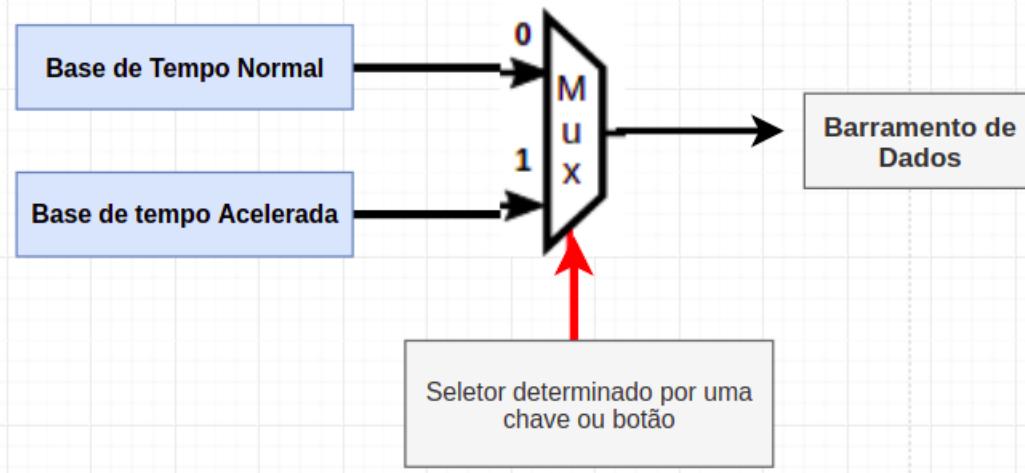


Figura 8: Mudança da base de tempo

Dessa forma, a seleção da base de tempo, é dado pelo seletor do MUX, que pode ser uma chave ou botão.

## 4 Mapa de Memoria

A tabela agora mostra a separação de endereços na memória RAM utilizado para guardar cada informação que permite o funcionamento do contador. De modo geral , podemos dizer que a divisão da memória ficou como abaixo:

1. Endereço 0 a 10 : Guarda constantes usadas.
2. Endereço 11 a 63 : Lixo de memória.

Exemplificando, a memória RAM ficou dividida no seguinte formato:

Endereço	Valor
0	0 (Constante)
1	1 (Constante)
2	2 (Constante)
3	3 (Constante)
4	4 (Constante)
5	5 (Constante)
6	6 (Constante)
7	-
8	-
9	9 (Constante)
10	10 (Constante)
11 - 63	-

Os registradores utilizados foram divididos da seguinte forma:

Registrador	Utilidade
0	Base de Tempo
1	Unidade segundo
2	Dezena segundo
3	Unidade minuto
4	Dezena minuto
5	Unidade hora
6	Dezena hora
7	Uso geral

O mapa de memória do projeto todo, pode ser representado pela figura abaixo.

Endereço em Decimal	Periférico
0-63	RAM
64-255	Reservado
256	LEDR0 - LEDR7
257	LEDR8
258	LEDR9
259-287	Reservado
288	HEX0
289	HEX1
290	HEX2
291	HEX3
292	HEX4
293	HEX5
294-319	Reservado
320	SW0-SW7
321	SW8
322	SW9 (Reservado para base de tempo)
323-351	Reservado
352	Bases de Tempo
353	KEY1
354	KEY2
355	KEY3
356	FPGA RESET
357-509	Reservado
510	Limpa Flipflop KEY1
511	Limpa FlipFlop da Base de tempo

---

## 5 Conceitos atingidos no Projeto

1. Assembler : Possibilitou a maior legibilidade e facilidade de construção do código. Sofreu mudanças na construção de suas instruções , devido a nova arquitetura (registrador-memória) e teve acréscimo de algumas instruções.
2. Configuração do horário atual : possibilidade de configurar o horário a qual deseja-se que o relógio comece a contar. Isso é feito por meio do botão KEY1 e pelo input das chaves da FPGA para inserir o valor.
3. Nova instrução (CLT) : Possibilita o desvio condicional caso o valor armazenado no registrador na instrução (Rx) possua valor menor que o valor armazenado em uma posição de memória. Essa instrução mostra-se útil quando desejamos impor um limite ao input do usuário,na hora da configuração do horario atual, por exemplo , impedindo que a unidade-segundos ultrapasse o valor 9 , criando um desvio condicional caso essa condição seja ou não verdade , setando uma flagMenor = 1, caso Rx < Mem[E].
4. Nova instrução (JLT) : Também utilizada na limitação do input do usuário na setagem do horario atual.Caso flagMenor = 1, realiza o desvio condicional para o endereço indicado na instrução.
5. Nova instrução (ADDI) : Realiza a soma do valor de Rx com o valor do imediato. Diferente da instrução anterior, que a soma era feita com um valor na posição de memória, essa instrução é mais rápida por não precisar realizar esse acesso.

## 6 Assembler

O código utilizado para criação do Assembler foi inspirado no criado pelo auxiliar da disciplina Design de computadores da instituição de ensino INSPER , Marco. O link do código do Marco pode ser encontrado [aqui](#).

O código do Assembler desenvolvido responsável pelo funcionamento do contador, encontra-se abaixo:

```
1
#
#
```

```
3 # INICIALIZAÇÃO DO RELÓGIO
# -----
5
# Zerando os displays de sete segmentos
7 .ZERA_DISPLAY
LDI R0 $0
STA @288 R0
STA @289 R0
11 STA @290 R0
STA @291 R0
13 STA @292 R0
STA @293 R0
15
# Apagando os LEDs
17 .APAGA_LEDs
LDI R0 $0
STA @256 R0
STA @257 R0
21 STA @258 R0
23
# Inicializando posicao de constantes imutáveis
25 LDI R0 $0
STA @0 R0
27
LDI R0 $1
29 STA @1 R0
31
LDI R0 $2
STA @2 R0
33
LDI R0 $5
35 STA @5 R0
37
LDI R0 $4
STA @4 R0
39
LDI R0 $9
```

```
41 STA @9 R0
43 LDI R0 $10
    STA @10 R0
45
    LDI R0 $6
47 STA @6 R0
49
    LDI R0 $3
    STA @3 R0
51
53 # Zerando registradores
    .ZERA_REGISTRADORES
55 LDI R0 $0
    LDI R1 $0
57 LDI R2 $0
    LDI R3 $0
59 LDI R4 $0
    LDI R5 $0
61 LDI R6 $0
    LDI R7 $0
63
# _____
65 # ROTINA PRINCIPAL
# _____
67
    .LOGICA_RELOGIO
69 NOP
    LDA R0 @352
71 OP_AND R0 @1
    CEQ R0 @1
73 JEQ .UNIDADE_SEGUNDO
    NOP
75 # Leitura KEY1 para colocar limite
    LDA R0 @353
77 OP_AND R0 @1
    CEQ R0 @1
```

```
79 JEQ .AJUSTAR_HORA
  NOP
81 JMP .LOGICA_RELOGIO

83 # ----- INCREMENTA RELÓGIO
# Soma unidade segundos
85 .UNIDADE_SEGUNDO
STA @511 R0
87 LDI R0 $0

89 CEQ R1 @9
JEQ .DEZENA_SEGUNDO

91 ADDI R1 $1
93 STA @288 R1
JMP .LOGICA_RELOGIO

95
# Soma dezena segundos
97 .DEZENA_SEGUNDO
LDI R1 $0
99 STA @288 R1

101 CEQ R2 @5
JEQ .UNIDADE_MINUTO

103 SOMA R2 @1
105 STA @289 R2
JMP .LOGICA_RELOGIO

107
# Soma unidade minutos
109 .UNIDADE_MINUTO
LDI R2 $0
111 STA @289 R2

113 CEQ R3 @9
JEQ .DEZENA_MINUTO

115 SOMA R3 @1
```

```
117 STA @290 R3
      JMP .LOGICA_RELOGIO

119
# Soma dezena minutos
121 .DEZENA_MINUTO
LDI R3 $0
STA @290 R3

125 CEQ R4 @5
JEQ .UNIDADE_HORA

127
SOMA R4 @1
129 STA @291 R4
JMP .LOGICA_RELOGIO

131
# Soma segundo horas
133 .UNIDADE_HORA
LDI R4 $0
STA @291 R4

137 CEQ R5 @9
JEQ .DEZENA_HORA

139
SOMA R5 @1
141 STA @292 R5

143 CEQ R5 @4
JEQ .LIMITE_MAXIMO

145 JMP .LOGICA_RELOGIO

147
# Soma dezena horas
149 .DEZENA_HORA
LDI R5 $0
STA @292 R5

151 SOMA R6 @1
STA @293 R6
```

```
155 JMP .LOGICA_RELOGIO

157 # ----- Limite máximo -----
    .LIMITE_MAXIMO

159 CEQ R6 @2
    JEQ .LIMITE_MAXIMO_ACONTEceu
161 JMP .LOGICA_RELOGIO

163
# ----- ACENDE LEDs ----- #
165 .LIMITE_MAXIMO_ACONTEceu
    LDI R0 $255
167 STA @256 R0
    LDI R0 $1
169 STA @257 R0
    STA @258 R0

171
    .ESPERA_RESET
173 LDA R0 @356
    OP_AND R0 @1
175 CEQ R0 @0
    JEQ .RESET
177 JMP .ESPERA_RESET

179 # ----- RESET ----- #
    .RESET

181 # Apaga LEDs
    LDI R0 $0
183 STA @256 R0
    STA @257 R0
185 STA @258 R0
    # Reseta REGISTRADORES
187 LDI R0 $0
    LDI R1 $0
189 LDI R2 $0
    LDI R3 $0
191 LDI R4 $0
    LDI R5 $0
```

```
193 LDI R6 $0
    LDI R7 $0
195 # Reseta displays
    LDI R0 $0
197 STA @288 R0
    STA @289 R0
199 STA @290 R0
    STA @291 R0
201 STA @292 R0
    STA @293 R0
203 JMP .LOGICA_RELOGIO

205 #
# AJUSTA HORA
207 #
# SETAR LIMITE
209 .AJUSTAR_HORA
    STA @510 R0
211
    LDI R0 $0
213 STA @288 R0
    STA @289 R0
215 STA @290 R0
    STA @291 R0
217 STA @292 R0
    STA @293 R0
219
# Leitura do limite de contagem unidade segundos
221 .LIMITE_UNIDADE_SEGUNDO
    LDI R0 $1
223 STA @256 R0

225 LDA R0 @320
    CLT R0 @10
227 JLT .VALOR_LIMITE_UNIDADE_SEGUNDO_OK
    JMP .VALOR_LIMITE_UNIDADE_SEGUNDO_ERRADO
229
    .VALOR_LIMITE_UNIDADE_SEGUNDO_OK
```

```

231 LDA R1 @320
    STA @288 R0
233 JMP .VALOR_LIMITE_UNIDADE_SEGUNDO_CONTINUA

235 .VALOR_LIMITE_UNIDADE_SEGUNDO_ERRADO
    LDI R1 $9
237 STA @288 R1
    JMP .VALOR_LIMITE_UNIDADE_SEGUNDO_CONTINUA
239
    .VALOR_LIMITE_UNIDADE_SEGUNDO_CONTINUA
241 LDA R0 @353
    OP_AND R0 @1
243 CEQ R0 @1
    JEQ .LIMITE_DEZENA_SEGUNDO_TEMP
245 JMP .LIMITE_UNIDADE_SEGUNDO

247
# Leitura do limite de contagem dezena segundos
249 .LIMITE_DEZENA_SEGUNDO_TEMP
    STA @510 R0
251
    .LIMITE_DEZENA_SEGUNDO
253 LDI R0 $2
    STA @256 R0
255
    LDA R0 @320
257 CLT R0 @5
    JLT .VALOR_LIMITE_DEZENA_SEGUNDO_OK
259 JMP .VALOR_LIMITE_DEZENA_SEGUNDO_ERRADO

261 .VALOR_LIMITE_DEZENA_SEGUNDO_OK
    LDA R2 @320
263 STA @289 R0
    JMP .VALOR_LIMITE_DEZENA_SEGUNDO_CONTINUA
265
    .VALOR_LIMITE_DEZENA_SEGUNDO_ERRADO
267 LDI R2 $5
    STA @289 R2

```

```
269 JMP .VALOR_LIMITE_DEZENA_SEGUNDO_CONTINUA  
  
271 .VALOR_LIMITE_DEZENA_SEGUNDO_CONTINUA  
LDA R0 @353  
OP_AND R0 @1  
CEQ R0 @1  
JEQ .LIMITE_UNIDADE_MINUTO_TEMP  
JMP .LIMITE_DEZENA_SEGUNDO  
  
277  
# Leitura do limite de contagem unidade minutos  
279 .LIMITE_UNIDADE_MINUTO_TEMP  
STA @510 R0  
  
281 .LIMITE_UNIDADE_MINUTO  
283 LDI R0 $4  
STA @256 R0  
  
285  
LDA R0 @320  
287 CLT R0 @9  
JLT .VALOR_LIMITE_UNIDADE_MINUTO_OK  
289 JMP .VALOR_LIMITE_UNIDADE_MINUTO_ERRADO  
  
291 .VALOR_LIMITE_UNIDADE_MINUTO_OK  
LDA R3 @320  
293 STA @290 R0  
JMP .VALOR_LIMITE_UNIDADE_MINUTO_CONTINUA  
  
295 .VALOR_LIMITE_UNIDADE_MINUTO_ERRADO  
297 LDI R3 $9  
STA @290 R3  
299 JMP .VALOR_LIMITE_UNIDADE_MINUTO_CONTINUA  
  
301 .VALOR_LIMITE_UNIDADE_MINUTO_CONTINUA  
LDA R0 @353  
303 OP_AND R0 @1  
CEQ R0 @1  
305 JEQ .LIMITE_DEZENA_MINUTO_TEMP  
JMP .LIMITE_UNIDADE_MINUTO
```

```
307 # Leitura do limite de contagem dezenas minutos
309 .LIMITE_DEZENA_MINUTO_TEMP
STA @510 R0
311
.LIMITE_DEZENA_MINUTO
313 LDI R0 $8
STA @256 R0
315
317 LDA R0 @320
CLT R0 @5
JLT .VALOR_LIMITE_DEZENA_MINUTO_OK
319 JMP .VALOR_LIMITE_DEZENA_MINUTO_ERRADO
321
321 .VALOR_LIMITE_DEZENA_MINUTO_OK
LDA R4 @320
323 STA @291 R0
JMP .VALOR_LIMITE_DEZENA_MINUTO_CONTINUA
325
325 .VALOR_LIMITE_DEZENA_MINUTO_ERRADO
327 LDI R4 $5
STA @291 R4
329 JMP .VALOR_LIMITE_DEZENA_MINUTO_CONTINUA
331
331 .VALOR_LIMITE_DEZENA_MINUTO_CONTINUA
LDA R0 @353
333 OP_AND R0 @1
CEQ R0 @1
335 JEQ .LIMITE_UNIDADE_HORA_TEMP
JMP .LIMITE_DEZENA_MINUTO
337
337 # Leitura do limite de contagem unidade horas
339 .LIMITE_UNIDADE_HORA_TEMP
STA @510 R0
341
341 .LIMITE_UNIDADE_HORA
343 LDI R0 $16
STA @256 R0
```

```
345 LDA R0 @320
347 CLT R0 @9
JLT .VALOR_LIMITE_UNIDADE_HORA_OK
349 JMP .VALOR_LIMITE_UNIDADE_HORA_ERRADO

351 .VALOR_LIMITE_UNIDADE_HORA_OK
LDA R5 @320
353 STA @292 R5
JMP .VALOR_LIMITE_UNIDADE_HORA_CONTINUA
355
.VALOR_LIMITE_UNIDADE_HORA_ERRADO
357 LDI R5 $9 # MEM[12] = valor setado unidade-hora
STA @292 R5
359 JMP .VALOR_LIMITE_UNIDADE_HORA_CONTINUA

361 .VALOR_LIMITE_UNIDADE_HORA_CONTINUA
LDA R0 @353
363 OP_AND R0 @1
CEQ R0 @1
365 JEQ .LIMITE_DEZENA_HORA_TEMP
JMP .LIMITE_UNIDADE_HORA

367
# Leitura do limite de contagem dezena horas
369 .LIMITE_DEZENA_HORA_TEMP
STA @510 R0

371
.LIMITE_DEZENA_HORA
373 LDI R0 $32
STA @256 R0

375
LDA R0 @320      # SW0-SW7
377
# Verifica unidade:
379 CLT R5 @4
JLT .DEZENA_HORA_LIMITE_2
381
# unidade-hora > 4 --> dezena-hora = 0 a 1
```

```
383 .DEZENA_HORA_LIMITE_1
CLT R0 @2 # r<2

385
JLT .VALOR_LIMITE_DEZENA_HORA_OK_1
JMP .VALOR_LIMITE_DEZENA_HORA_ERRADO_1

387
389 # ----- Dezena hora ok 1
.Valor_Limite_Dezena_Hora_OK_1
391 LDA R6 @320
STA @293 R0
393 JMP .VALOR_LIMITE_DEZENA_HORA_CONTINUA

395 # ----- Dezena hora errado 1
.Valor_Limite_Dezena_Hora_ERRADO_1
397 LDI R6 $1
STA @293 R6
399 JMP .VALOR_LIMITE_DEZENA_HORA_CONTINUA

401 # unidade-hora < 4 --> dezena-hora = 0 a 2
.DEZENA_HORA_LIMITE_2
403 CLT R0 @3

405 JLT .VALOR_LIMITE_DEZENA_HORA_OK_2
JMP .VALOR_LIMITE_DEZENA_HORA_ERRADO_2

407
409 # ----- Dezena hora ok 2
.Valor_Limite_Dezena_Hora_OK_2
411 LDA R6 @320
STA @293 R0
413 JMP .VALOR_LIMITE_DEZENA_HORA_CONTINUA

415 # ----- Dezena hora errado 2
.Valor_Limite_Dezena_Hora_ERRADO_2
417 LDI R6 $2
STA @293 R6
419 JMP .VALOR_LIMITE_DEZENA_HORA_CONTINUA

419 # ----- LOOP KEY1
```

```
421 .VALOR_LIMITE_DEZENA_HORA_CONTINUA
LDA R0 @353
423 OP_AND R0 @1
CEQ R0 @1
425 JEQ .FIM_LIMITE
JMP .LIMITE_DEZENA_HORA
427
# ----- FIM AJUSTE
429 .FIM_LIMITE
STA @510 R0
431 LDI R0 $0
STA @256 R0
433 STA @511 R0
JMP .LOGICA_RELOGIO
```

## 7 Link do vídeo

Caso queira visualizar o funcionamento do programa , accesse o [link](#).

## Referências

- [1] Disciplina Design de Computadores - INSPER 2023. Accessed: 2022-04-24.