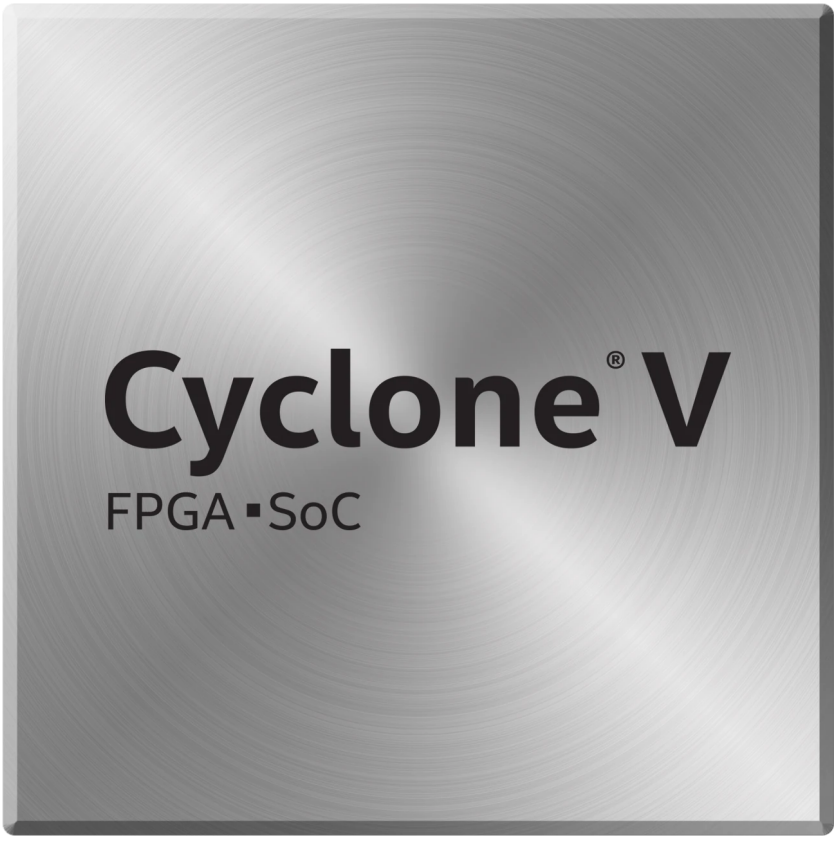
Contador V0



Caio Travain

André Brito

16.10.2023

DESIGN DE COMPUTADORES

INTRODUÇÃO

O projeto consiste em um contador onde podemos utilizar de diversas funcionalidades. Como configuração de um alvo para o contador, fazendo com que avise quando esse falor for atingido. Adição e Subtração, mostrando toda versatilidade e avisando qual modo está através do LED9. Além disso, conta com um botão de zerar contagem. O contador utiliza uma arquitetura Registrador-Memória.

UTILIZAÇÃO

Para utilizar podemos clicar em KEY0 para adicionar um valor ao contador, e pode ir somando até o valor 999.999. Ao pressionar KEY1 você abre a configuração do alvo de contagem, assim você pode clicar em KEY0 para ir trocando o valor em cada casa decimal e trocar apertando KEY1 (A casa decimal que está sendo configurada fica piscando), para voltar ao contador basta acabar a configuração da última casa decimal e clicar em KEY1. Para utilizar a função de contagem regressiva, basta subir a chave SW9 e o LED9 irá acender, agora se clicar em KEY0 irá subtrair valor da contagem. Caso chegue no valor alvo, todos os leds se acenderam e a contagem será travada. Para zerar a contagem, basta clicar no botão FPGA_RESET e o contador irá zerar.

Características

1. Arquitetura Registrador-Memória (4 registradores)
2. Instruções (Utilização em negrito):
    a. **LDI $(*VALOR*), REG(*Endereço do registrador*)** -Carrega o*VALOR* no Registardor
    b. **LDA @(*Endereço de Memória*), REG(*Endereço do registrador*)** - Carrega o valor *Endereço de Memória* no Registrador Escolhido
    c. **SOMA @(*Endereço de Memória*), REG(*Endereço do registrador*)** - Soma o valor *Endereço de Memória e o registrador Escolhido* e salva no Registrador Escolhido
    d. **SUB@(*Endereço de Memória*), REG(*Endereço do registrador*)** - Subtrai o valor *Endereço de Memória e o registrador Escolhido* e salva no Registrador Escolhido
    e. **STA@(*Endereço de Memória*), REG(*Endereço do registrador*)** - Salva o

valor do *registrador Escolhido* no *Endereço de Memória*

f. **JMP @(*Endereço de destino) ou JMP %(label)*** - vai para a linha do *Endereço de destino ou da label*

g. **JEQ@(*Endereço de destino) ou JEQ%(label)*** - vai para a linha do *Endereço de destino ou da label* caso a flag igual esteja acionada

h. ***CEQ*@(*Endereço de Memória*), REG(*Endereço de Memória*)** - Compara o valor do *Endereço de Memória e do Endereço de Memória* e ativar a flag_igual caso seja o mesmo número.

i. **JSR@(*Endereço de subrotina) ou JSR%(label)*** - vai para a linha do *Endereço de destino ou da label* e salva o endereço de retorno

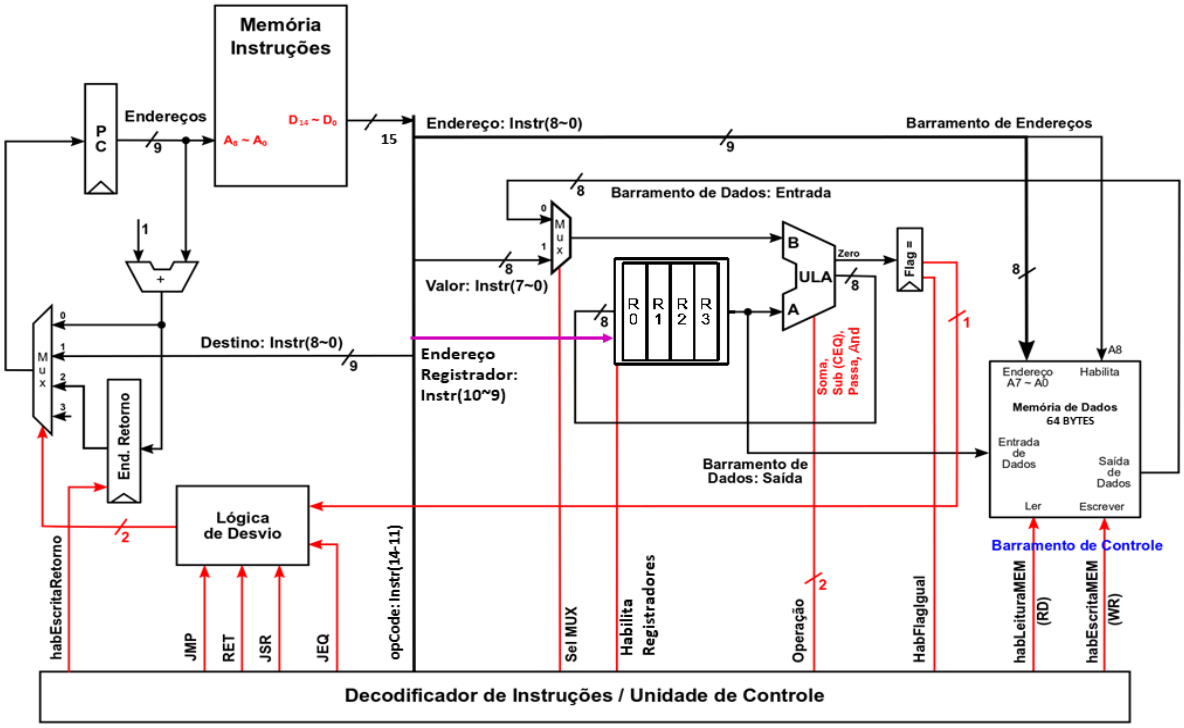
j. **RET** - Retorna de uma sub-rotina

k. **OP_AND@(*Endereço de Memória*), REG(*Endereço do registrador*)** - Faz operação AND com o valor *Endereço de Memória e o registrador Escolhido* e salva no Registrador Escolhido

l. **NOP** - Faz Nada

3. Formato das instruções (15 bits):
    a. **(14 ~ 11) *Opcode***, que será transformado nos sinais de controle
    b. **(10 ~ 9) *Endereço do Registrador***, que dirá o registrador usado, caso não seja especificado na instrução, será usado o registrador 0
    c. **(8 ~ 0) *Endereço RAM* ou *Destino Jump,*** dependendo da instrução será o endereço da RAM ou o endereço de destino do JUMP
        i. **(7 ~ 0) *Valor***, os oito bits menos significativos podem também ter o valor do imediato dependendo da instrução

FLUXO DE DADOS PROCESSADOR



Aqui foi apresentado o fluxo de dados da CPU com as memórias ROM e RAM. O diagrama mostra a implementação da operação AND, do endereço de registrador da arquitetura Registrador-Memória e o bloco de registradores. Além disso, vale notar o tamnho da instrução, que é de 15 bits diferente do realizado em aula.

Pontos de Controle

1. Listado a seguir os pontos de controle para cada instrução
    a. (Operação ULA => “01” - Soma, “10” - And, “11” - Sub, “00” - Passa)

Instrução	Habilita Retorno	JMP	RET	JSR	JEQ	Seletor Mux	Habilita Reg	Operação ULA (2 bits)	Habilita Flag Zero	RD	WR
LDI	0	0	0	0	0	1	1	00	0	0	0
LDA	0	0	0	0	0	0	1	00	0	1	0
SOMA	0	0	0	0	0	0	1	01	0	1	0
SUB	0	0	0	0	0	0	1	11	0	1	0
STA	0	0	0	0	0	0	0	00	0	0	1
JMP	0	1	0	0	0	0	0	00	0	0	0
JEQ	0	0	0	0	1	0	0	00	0	0	0
CEQ	0	0	0	0	0	0	0	11	1	1	0
JSR	1	0	0	1	0	0	0	00	0	0	0
RET	0	0	1	0	0	0	0	00	0	0	0
OP_AND	0	0	0	0	0	0	1	10	0	1	0
NOP	0	0	0	0	0	0	0	00	0	0	0

Mapa de Memória

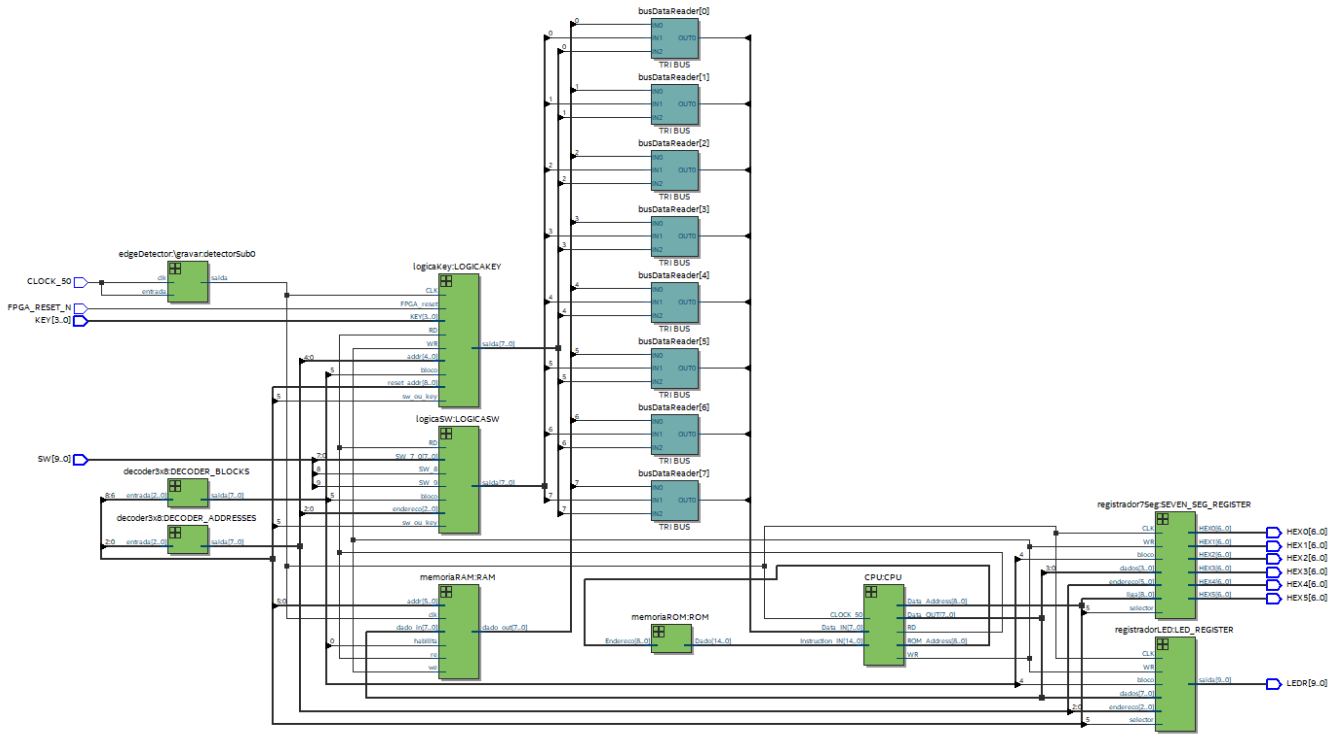
Endereço	Função	Variavéis
0 - 63 (Read and Write)	Memória Ram utilizavél para o programa	0 a 5 - Valores da unidade, dezena, centena, milhar, dezena de milhar e centena de milhar respectivamente. 10 a 15 - Valores dos limites unidade, dezena, centena, milhar, dezena de milhar e centena de milhar respectivamente. 51 , 59 e 60 - Valores utilizados para comparações, sendo eles 1, 9 e 10 respectivamente. 29 - Valor de temporização utilizado para piscar o display, sendo ele 99 20, 21, 22 - valores dos temporizadores, sendo valores que quando atingir 100 aumenta o valor do próximo. 23 - Variavél PISCA, define se vai estar ligado ou desligado o display enquanto estiver piscando
256 - 258 (Write)	LEDS	256 - Leds 7 até 0, o valor define quais estão ligados ou desligados (8 bits) 257- Led 8, ‘1’ ligado e ‘0’ desligado (1bit) 256 - Led 9, ‘1’ ligado e ‘0’

		desligado (1 bit)
288 - 293 (Write)	Displays de sete segmentos (4 bits)	288 - 293, configuram o valor dos displays HEX 0, HEX 1, HEX 2, HEX 3, HEX 4 e HEX 5 respectivamente.
320 - 322 (Read)	Chaves SW	320 - Lê os valores de SW7 até SW0 (8 bits) 321 - Lê os valores de SW8 (1 bit) 322 - Lê os valores de SW9 (1bit)
352 - 356 (Read)	Botões KEYS (1bit)	352 a 356 são os valores de KEY0, KEY1, KEY2, KEY3 E FPGA_RESET respectivamente.
500 - 505 (Write)	Desligar os Displays de sete segmentos (1 bit)	500 a 505, quando ‘1’ desligam o HEX 0, HEX 1, HEX 2, HEX 3, HEX 4 e HEX 5 respectivamente e ‘0’ liga.
511 - 510 (Write)	Utilizado para limpar valores de botões (1 bit)	511- limpa o valor de KEY0 510 - limpa o valor de KEY1

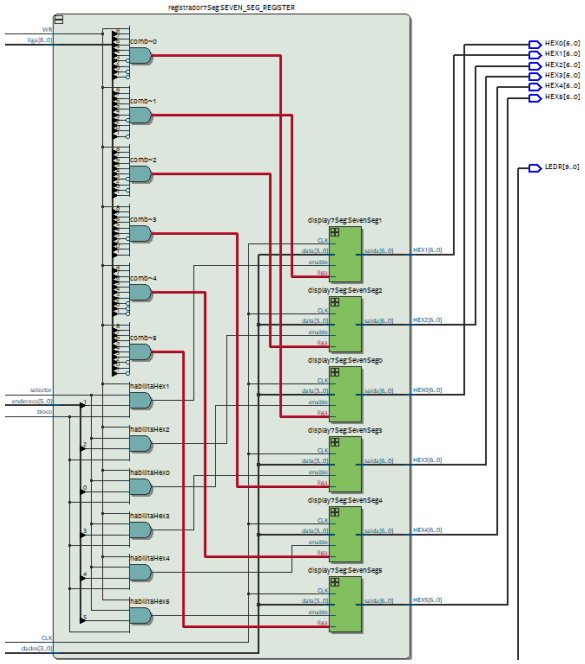
I/O Pins

1. **KEY 0**
    a. Utilizado para aumentar ou subtrair o valor de contagem
    b. Quando em modo de configuração de número alvo aumenta a casa decimal
2. **KEY 1**
    a. Entra em modo de configuração de limite, a cada clique muda a casa decimal até a centenas de milhares depois sai do modo de configuração
3. **FPGA RESET**
    a. Limpa a contagem mantendo o limite ja configurado
4. **SW 9**
    a. Ativa o modo de subtração do contador, não é válido no modo de configuração de valor alvo

RTL

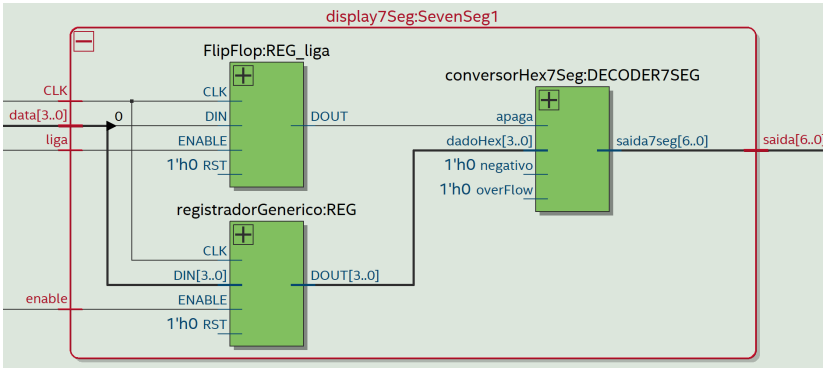


Adicionando Logica para Desligar os Displays Sete Segmentos



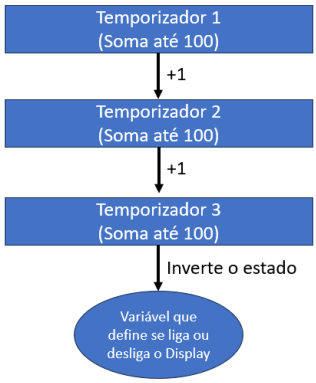
Foi adicionado uma entrada dos endereços que sai da CPU, na conexão chamada “liga”, assim verificamos usando um AND se os endereços são 500, 501, 502, 503, 504 ou 505. Caso seja um desses e o sinal de “*Write*” estiver ativado, desligue o Display caso tenha o valor 1 e liga caso esteja com valor 0, sendo HEX 0 ~ 500 à HEX 5 ~ 505.

Exemplo dentro dos componentes “*DisplaySeg*” :



Onde vemos que temos o registrador do valor do registrador com o que aparece no display e um FlipFlop que salva o valor se vai deixar ligado ou desligado o Display.

Foi utilizado esse sinal para manter a casa decimal que está sendo mudada na configuração de valor alvo piscando. Para isso realizamos três temporizadores em software que conta toda vez que sua subrotina é chamada e armazena tal valor, assim quando chegar em 100, ele atualiza outro soma um em outro temporizador e quando o segundo temporizador chega a 100, adiciona um em um outro temporizador. Assim quando o último chegar em 100, temos que aconteceram 1.000.000 chamadas, depois trocamos o valor da variável em memória que decide se o display vai estar ligado ou desligado. Foi usado essa valor em uma estimativa, baseando-se do clock que é de 50Mhz, realizando 50M intruções por segundo, como temos outras intruções entre a chamada da subrotina para piscar, utilizamos o valor 1M que apresentou uma frequência decente e utilizavél. Foi contado que havia entre 20 a 40 intruções entre a contagem do temporizador dependendo do estado do contador, ou seja, o display ficaria ligado entre 0,5 - 0,8 segundos e depois trocaria seu estado.



ASSEMBLER

Foi criado um assembler que formata as instruções abaixo em linguagem de máquina. Usando como base o assembler do Marco Mello e otimizado para remover as linhas vazias e adicionar a opção de usar labels para fazer pulos. Também foi adicionado o endereçamento do registrador e caso não seja especificado, utilizará o registrador 0. Devido ao corte de linhas em brancos, mesmo que o código a seguir passe de 512 linhas, podemos utilizar pelo fato que muitas são linhas em branco usadas para organização. Também adicionamos a operação AND no assembler.

ASSEMBLY CODE

1. RESET:
2. STA @511 # Limpando key 0
3. STA @510 # Limpando key 1
4. STA @509 # Limpando reset_key
5. STA @0 # Limpando endereço de unidade
6. STA @1 # Limpando endereço de dezena
7. STA @2 # Limpando endereço de centena
8. STA @3 # Limpando endereço de milhar
9. STA @4 # Limpando endereço de dezena de milhar
10. STA @5 # Limpando endereço de centena de milhar
11. STA @288 # Limpando endereço do HEX0
12. STA @289 # Limpando endereço do HEX1
13. STA @290 # Limpando endereço do HEX2
14. STA @291 # Limpando endereço do HEX3
15. STA @292 # Limpando endereço do HEX4
16. STA @293 # Limpando endereço do HEX5
17. LDI $9 # Carregando 9 no acumulador
18. STA @59 # Carregando 9 na posição 59
19. LDI $10 # Carregando 10 no acumulador
20. STA @60 # Carergando 10 na posição 60
21. LDI $1 # Carregando 1 no acumulador
22. STA @51 # Carregando 1 na posição 51
23. LDI $9
24. STA @10 # Limite das unidades
25. STA @11 # Limite das dezenas
26. STA @12 # Limite das centenas
27. STA @13 # Limite dos milhares
28. STA @14 # Limite das dezenas de milhares
29. STA @15 # Limite das centenas de milhares
30. LDI $99 # Carregando 99 no acumulador
31. STA @29 # Carregando 100 na posição 29
32. LDI $0 # Carregando 0 no acumulador
33. STA @500 # Desligando o display 0
34. STA @20 #temporizador 1

35. STA @21 #temporizador 2
36. STA @22 #temporizador 3
37. STA @23 # PISCA OU NÃO PISCA
38. JMP %le_key # Vai para o label le_key
39.
40. temporizador_1_segundo:
41. LDA @20 # Carrega o acumulador com o endereço de temporizador 1
42. CEQ @29 # Compara o valor do acumulador com o valor 99
43. JEQ %temporizador_2_segundo # Se for igual, vai para o label temporizador_2_segundo
44. SOMA @51 # Soma 1 no acumulador
45. STA @20 # Armazena o valor do acumulador no endereço de temporizador 1
46. RET
47.
48. temporizador_2_segundo:
49. LDI $0 # Carrega 0 no acumulador
50. STA @20 # Armazena o valor do acumulador no endereço de temporizador 1
51. LDA @21 # Carrega o acumulador com o endereço de temporizador 2
52. CEQ @29 # Compara o valor do acumulador com o valor 99
53. JEQ %temporizador_3_segundo # Se for igual, vai para o label temporizador_3_segundo
54. SOMA @51 # Soma 1 no acumulador
55. STA @21 # Armazena o valor do acumulador no endereço de temporizador 2
56. RET
57.
58. temporizador_3_segundo:
59. LDI $0 # Carrega 0 no acumulador
60. STA @21 # Armazena o valor do acumulador no endereço de temporizador 2
61. LDA @22 # Carrega o acumulador com o endereço de temporizador 3
62. CEQ @29 # Compara o valor do acumulador com o valor 99
63. JEQ %PISCA # Se for igual, vai para o label LIMPA
64. SOMA @51 # Soma 1 no acumulador
65. STA @22 # Armazena o valor do acumulador no endereço de temporizador 3
66. RET
67.
68. PISCA:
69. LDI $0 # Carrega 0 no acumulador

70. STA @22 # Limpando temporizador 3
71. LDA @23 # Carrega o acumulador com o endereço de PISCA
72. CEQ @51 # Compara o valor do acumulador com o valor 1
73. JEQ %LIMPA # Se for igual, vai para o label LIMPA
74. LDI $1 # Carrega 1 no acumulador
75. STA @23 # Armazena o valor do acumulador no endereço de PISCA
76. RET
77.
78. LIMPA:
79. LDI $0 # Carrega 0 no acumulador
80. STA @23 # Armazena o valor do acumulador no endereço de PISCA
81. RET
82.
83. define_limites_unidades:
84. STA @510 # Limpando key 1
85. STA @511 # Limpando key 0
86. JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
87.
88. checa_limites_unidades:
89. JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
90. LDA @23 # Carrega o acumulador com o endereço de PISCA
91. STA @500 # Desliga o display unidade
92. LDA @353 # Carrega o acumulador com o key 1
93. CEQ @51 # Compara o valor do acumulador com o valor 1
94. JEQ %define_limites_dezenas # Se for igual, vai para o label define_limites_dezenas
95. LDA @352 # Carrega o acumulador com o key 0
96. CEQ @50 # Compara o valor do acumulador com o valor 0
97. JEQ %checa_limites_unidades # Se for igual, vai para o label checa_limites_unidades
98. JSR %adiciona_unidade # Se não for igual, vai para o label adiciona_unidade
99. JMP %define_limites_unidades # Se não for igual, volta para o label define_limites_unidades
100.
101.
102.
103. adiciona_unidade:

```
104.    STA @511 # Limpando key 0
105.    LDA @10 # Carrega o acumulador com o endereço de Limite de unidade
106.    CEQ @59 # Compara o valor do acumulador com o valor 9
107.    JEQ %zera_unidade # Se for igual, vai para o label zera_unidade
108.    SOMA @51 # Soma 1 no acumulador
109.    STA @10 # Armazena o valor do acumulador no endereço de Limite de
        unidade
110.    RET
111.    zera_unidade:
112.    LDI $0 # Carrega 0 no acumulador
113.    STA @10 # Armazena o valor do acumulador no endereço de Limite de
        unidade
114.    RET
115.
116.    define_limites_dezenas:
117.    STA @510 # Limpando key 1
118.    STA @511 # Limpando key 0
119.    JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
120.
121.    checa_limites_dezenas:
122.    LDI $0 # Carrega 0 no acumulador
123.    STA @500 # liga o display unidade
124.    JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
125.    LDA @23 # Carrega o acumulador com o endereço de PISCA
126.    STA @501 # Desliga o display unidade
127.    LDA @353 # Carrega o acumulador com o key 1
128.    CEQ @51 # Compara o valor do acumulador com o valor 1
129.    JEQ %define_limites_centenas # Se for igual, vai para o label
        define_limites_dezenas
130.    LDA @352 # Carrega o acumulador com o key 0
131.    CEQ @50 # Compara o valor do acumulador com o valor 0
132.    JEQ %checa_limites_dezenas # Se for igual, vai para o label
        checa_limites_dezenas
133.    JSR %adiciona_dezena # Se não for igual, vai para o label adiciona_dezena
134.    STA @11 # Armazena o valor do acumulador no endereço de Limite de dezenas
135.    JMP %define_limites_dezenas # Se não for igual, volta para o label
        define_limites_dezenas
```

```
136.
137.    adiciona_dezena:
138.    STA @511 # Limpando key 0
139.    LDA @11 # Carrega o acumulador com o endereço de Limite de dezena
140.    CEQ @59 # Compara o valor do acumulador com o valor 9
141.    JEQ %zera_dezena # Se for igual, vai para o label zera_dezena
142.    SOMA @51 # Soma 1 no acumulador
143.    RET
144.    zera_dezena:
145.    LDI $0 # Carrega 0 no acumulador
146.    RET
147.
148.    define_limites_centenas:
149.    STA @510 # Limpando key 1
150.    STA @511 # Limpando key 0
151.    JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
152.
153.    checa_limites_centenas:
154.    LDI $0 # Carrega 0 no acumulador
155.    STA @501 # liga o display unidade
156.    JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
157.    LDA @23 # Carrega o acumulador com o endereço de PISCA
158.    STA @502 # Desliga o display unidade
159.    LDA @353 # Carrega o acumulador com o key 1
160.    CEQ @51 # Compara o valor do acumulador com o valor 1
161.    JEQ %define_limites_milhares # Se for igual, vai para o label
    define_limites_centenas
162.    LDA @352 # Carrega o acumulador com o key 0
163.    CEQ @50 # Compara o valor do acumulador com o valor 0
164.    JEQ %checa_limites_centenas # Se for igual, vai para o label
    checa_limites_centenas
165.    JSR %adiciona_centena # Se não for igual, vai para o label adiciona_centena
166.    STA @12 # Armazena o valor do acumulador no endereço de Limite de
    centenas
167.    JMP %define_limites_centenas # Se não for igual, volta para o label
    define_limites_centenas
168.
```

```
169.  adiciona_centena:
170.  STA @511 # Limpando key 0
171.  LDA @12 # Carrega o acumulador com o endereço de Limite de centena
172.  CEQ @59 # Compara o valor do acumulador com o valor 9
173.  JEQ %zera_centena # Se for igual, vai para o label zera_centena
174.  SOMA @51 # Soma 1 no acumulador
175.  RET
176.  zera_centena:
177.  LDI $0 # Carrega 0 no acumulador
178.  RET
179.
180.  define_limites_milhares:
181.  STA @510 # Limpando key 1
182.  STA @511 # Limpando key 0
183.  JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
184.
185.  checa_limites_milhares:
186.  LDI $0 # Carrega 0 no acumulador
187.  STA @502 # liga o display unidade
188.  JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
189.  LDA @23 # Carrega o acumulador com o endereço de PISCA
190.  STA @503 # Desliga o display unidade
191.  LDA @353 # Carrega o acumulador com o key 1
192.  CEQ @51 # Compara o valor do acumulador com o valor 1
193.  JEQ %define_limites_dezenas_de_milhares # Se for igual, vai para o label
    define_limites_milhares
194.  LDA @352 # Carrega o acumulador com o key 0
195.  CEQ @50 # Compara o valor do acumulador com o valor 0
196.  JEQ %checa_limites_milhares # Se for igual, vai para o label
    checa_limites_milhares
197.  JSR %adiciona_milhar # Se não for igual, vai para o label adiciona_milhar
198.  STA @13 # Armazena o valor do acumulador no endereço de Limite de
    milhares
199.  JMP %define_limites_milhares # Se não for igual, volta para o label
    define_limites_milhares
200.
201.  adiciona_milhar:
```

```
202.    STA @511 # Limpando key 0
203.    LDA @13 # Carrega o acumulador com o endereço de Limite de milhar
204.    CEQ @59 # Compara o valor do acumulador com o valor 9
205.    JEQ %zera_milhar # Se for igual, vai para o label zera_milhar
206.    SOMA @51 # Soma 1 no acumulador
207.    RET
208.    zera_milhar:
209.    LDI $0 # Carrega 0 no acumulador
210.    RET
211.
212.    define_limites_dezenas_de_milhares:
213.    STA @510 # Limpando key 1
214.    STA @511 # Limpando key 0
215.    JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
216.
217.    checa_limites_dezenas_de_milhares:
218.    LDI $0 # Carrega 0 no acumulador
219.    STA @503 # liga o display unidade
220.    JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
221.    LDA @23 # Carrega o acumulador com o endereço de PISCA
222.    STA @504 # Desliga o display unidade
223.    LDA @353 # Carrega o acumulador com o key 1
224.    CEQ @51 # Compara o valor do acumulador com o valor 1
225.    JEQ %define_limites_centenas_de_milhares # Se for igual, vai para o label
    define_limites_dezenas_de_milhares
226.    LDA @352 # Carrega o acumulador com o key 0
227.    CEQ @50 # Compara o valor do acumulador com o valor 0
228.    JEQ %checa_limites_dezenas_de_milhares # Se for igual, vai para o label
    checa_limites_dezenas_de_milhares
229.    JSR %adiciona_dezena_de_milhar # Se não for igual, vai para o label
    adiciona_dezena_de_milhar
230.    STA @14 # Armazena o valor do acumulador no endereço de Limite de dezenas
    de milhares
231.    JMP %define_limites_dezenas_de_milhares # Se não for igual, volta para o label
    define_limites_dezenas_de_milhares
232.
233.    adiciona_dezena_de_milhar:
```

```
234.    STA @511 # Limpando key 0
235.    LDA @14 # Carrega o acumulador com o endereço de Limite de dezena de
    milhar
236.    CEQ @59 # Compara o valor do acumulador com o valor 9
237.    JEQ %zera_dezena_de_milhar # Se for igual, vai para o label
    zera_dezena_de_milhar
238.    SOMA @51 # Soma 1 no acumulador
239.    RET
240.    zera_dezena_de_milhar:
241.    LDI $0 # Carrega 0 no acumulador
242.    RET
243.
244.    define_limites_centenas_de_milhares:
245.    STA @510 # Limpando key 1
246.    STA @511 # Limpando key 0
247.    JSR %atualiza_displays_limites # Vai para o label atualiza_displays_limites
248.
249.    checa_limites_centenas_de_milhares:
250.    LDI $0 # Carrega 0 no acumulador
251.    STA @504 # liga o display unidade
252.    JSR %temporizador_1_segundo # Vai para o label temporizador_1_segundo
253.    LDA @23 # Carrega o acumulador com o endereço de PISCA
254.    STA @505 # Desliga o display unidade
255.    LDA @353 # Carrega o acumulador com o key 1
256.    CEQ @51 # Compara o valor do acumulador com o valor 1
257.    JEQ %pre_le_key # Se for igual, vai para o label pre_le_key
258.    LDA @352 # Carrega o acumulador com o key 0
259.    CEQ @50 # Compara o valor do acumulador com o valor 0
260.    JEQ %checa_limites_centenas_de_milhares # Se for igual, vai para o label
    checa_limites_centenas_de_milhares
261.    JSR %adiciona_centena_de_milhar # Se não for igual, vai para o label
    adiciona_centena_de_milhar
262.    STA @15 # Armazena o valor do acumulador no endereço de Limite de
    centenas de milhares
263.    JMP %define_limites_centenas_de_milhares # Se não for igual, volta para o
    label define_limites_centenas_de_milhares
264.
```

```
265.  adiciona_centena_de_milhar:
266.  STA @511 # Limpando key 0
267.  LDA @15 # Carrega o acumulador com o endereço de Limite de centena de
      milhar
268.  CEQ @59 # Compara o valor do acumulador com o valor 9
269.  JEQ %zera_centena_de_milhar # Se for igual, vai para o label
      zera_centena_de_milhar
270.  SOMA @51 # Soma 1 no acumulador
271.  RET
272.  zera_centena_de_milhar:
273.  LDI $0 # Carrega 0 no acumulador
274.  RET
275.
276.  pre_le_key:
277.  STA @511 # Limpando key 0
278.  STA @510 # Limpando key 1
279.  LDI $0 # Carrega 1 no acumulador
280.  STA @505 # Liga o display 0
281.  JMP %atualiza_displays
282.  JMP %le_key # Vai para o label le_key
283.  le_key:
284.  LDA @352 # Carrega o acumulador com o key 0
285.  CEQ @51 # Compara o valor do acumulador com o valor 1
286.  JEQ %incrementa_unidade # Se for igual, vai para o label incrementa
287.  LDA @353 # Carrega o acumulador com o key 1
288.  CEQ @51 # Compara o valor do acumulador com o valor 1
289.  JEQ %define_limites_unidades # Se for igual, vai para o label incrementa
290.  JSR %checa_op # checa se é + ou -
291.  LDA @356# Carrega o acumulador com o endereço de fpga_reset
292.  CEQ @51 # Compara o valor do acumulador com o valor 1
293.  JEQ %RESET_FPGA # Se for igual, vai para o label RESET_FPGA
294.  JMP %le_key # Se não for igual, volta para o label le_key
295.
296.  checa_op:
297.  LDA @322 # Carrega o acumulador com o endereço de SW9
298.  CEQ @50 # Compara o valor do acumulador com o valor 0
299.  JEQ %mais # Se for igual, vai para o label le_key
```

```
LDI $1 # Carrega o acumulador com o valor 1
STA @258 # Liga led 9
RET
mais:
LDI $0 # Carrega o acumulador com o valor 1
STA @258 # Liga led 9
RET

incrementa_unidade:
STA @511 # Limpando key 0
LDA @322 # Carrega o acumulador com o endereço de SW9
CEQ @51 # Compara o valor do acumulador com o valor 1
JEQ %sub_unidade # Se for igual, vai para o label sub_unidade
LDA @0 # Carrega o acumulador com o endereço de unidade
CEQ @59 # Compara o valor do acumulador com o valor 9
JEQ %incrementa_dezena # Se for igual, vai para o label incrementa_dezena
SOMA @51 # Soma 1 no acumulador
STA @0 # Armazena o valor do acumulador no endereço de unidade
LDI $0 # Carrega o acumulador com o valor 1
STA @258 # Desliga led 9
JMP %atualiza_displays # Se não for igual, vai para o label atualiza_unidade
sub_unidade:
LDA @0 # Carrega o acumulador com o endereço de unidade
CEQ @50 # Compara o valor do acumulador com o valor 0
JEQ %decremeta_dezena # Se for igual, vai para o label decremeta_dezena
SUB @51 # Subtrai 1 no acumulador
STA @0 # Armazena o valor do acumulador no endereço de unidade
JMP %atualiza_displays # Se não for igual, vai para o label atualiza_unidade

incrementa_dezena:
LDI $0
STA @0 # Limpando endereço de unidade
STA @511 # Limpando Key 0
LDA @1 # Carregar o acumulador com o endereço da dezena
CEQ @59 # Compara o valor do acumulador (dezena) com o valor 9
JEQ %incrementa_centena # Se for igual a 9, vai para o incrementa_centena
SOMA @51 # Soma 1 no acumulador
```

```
STA @1 # Armazena o valor do acumulador no endereço das dezenas
JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays

decremeta_dezena:
LDI $9
STA @0 # Limpando endereço de unidade
STA @511 # Limpando Key 0
LDA @1 # Carregar o acumulador com o endereço da dezena
CEQ @50 # Compara o valor do acumulador (dezena) com o valor 0
JEQ %decremeta_centena # Se for igual a 0, vai para o decremeta_centena
SUB @51 # Subtrai 1 no acumulador
STA @1 # Armazena o valor do acumulador no endereço das dezenas
JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays

incrementa_centena:
LDI $0
STA @1 # Limpando endereço de dezena
STA @511 # Limpando Key 0
LDA @2 # Carregar o acumulador com o endereço da centena
CEQ @59 # Compara o valor do acumulador (centena) com o valor 9
JEQ %incrementa_milhar # Se for igual a 9, vai para o incrementa_milhar
SOMA @51 # Soma 1 no acumulador
STA @2 # Armazena o valor do acumulador no endereço das centenas
JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays

decremeta_centena:
LDI $9
STA @1 # Limpando endereço de dezena
STA @511 # Limpando Key 0
LDA @2 # Carregar o acumulador com o endereço da centena
CEQ @50 # Compara o valor do acumulador (centena) com o valor 0
JEQ %decremeta_milhar # Se for igual a 0, vai para o decremeta_milhar
SUB @51 # Subtrai 1 no acumulador
STA @2 # Armazena o valor do acumulador no endereço das centenas
JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays

incrementa_milhar:
```

```
374.    LDI $0
375.    STA @2 # Limpando endereço de centena
376.    STA @511 # Limpando Key 0
377.    LDA @3 # Carregar o acumulador com o endereço da milhares
378.    CEQ @59 # Compara o valor do acumulador (dezena) com o valor 9
379.    JEQ %incrementa_dezena_de_milhar # Se for igual a 9, vai para o
    incrementa_dezena_de_milhar
380.    SOMA @51 # Soma 1 no acumulador
381.    STA @3 # Armazena o valor do acumulador no endereço das milhares
382.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_unidade
383.
384.    decremeta_milhar:
385.    LDI $9
386.    STA @2 # Limpando endereço de centena
387.    STA @511 # Limpando Key 0
388.    LDA @3 # Carregar o acumulador com o endereço da milhares
389.    CEQ @50 # Compara o valor do acumulador (dezena) com o valor 0
390.    JEQ %decremeta_dezena_de_milhar # Se for igual a 0, vai para o
    decremeta_dezena_de_milhar
391.    SUB @51 # Subtrai 1 no acumulador
392.    STA @3 # Armazena o valor do acumulador no endereço das milhares
393.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_unidade
394.
395.    incrementa_dezena_de_milhar:
396.    LDI $0
397.    STA @3 # Limpando endereço de milhar (RAM 3)
398.    STA @511 # Limpando Key 0
399.    LDA @4 # Carregar o acumulador com o endereço da dezena de milhar
400.    CEQ @59 # Compara o valor do acumulador (dezena) com o valor 9
401.    JEQ %incrementa_centena_milhar # Se for igual a 9, vai para o
    incrementa_centena_milhar
402.    SOMA @51 # Soma 1 no acumulador
403.    STA @4 # Armazena o valor do acumulador no endereço das milhares
404.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays
405.
406.    decremeta_dezena_de_milhar:
407.    LDI $9
```

```
408.    STA @3 # Limpando endereço de milhar (RAM 3)
409.    STA @511 # Limpando Key 0
410.    LDA @4 # Carregar o acumulador com o endereço da dezena de milhar
411.    CEQ @50 # Compara o valor do acumulador (dezena) com o valor 0
412.    JEQ %decremeta_centena_milhar # Se for igual a 0, vai para o
    decremeta_centena_milhar
413.    SUB @51 # Subtrai 1 no acumulador
414.    STA @4 # Armazena o valor do acumulador no endereço das milhares
415.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays
416.
417.    incrementa_centena_milhar:
418.    LDI $0
419.    STA @4 # Limpando endereço de dezena de milhar
420.    STA @511 # Limpando Key 0
421.    LDA @5 # Carregar o acumulador com o endereço da centena
422.    CEQ @59 # Compara o valor do acumulador (centena) com o valor 9
423.    JEQ %atualiza_displays # Se for igual a 9, vai para o atualiza_displays
424.    SOMA @51 # Soma 1 no acumulador
425.    STA @5 # Armazena o valor do acumulador no endereço das centenas
426.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays
427.
428.    decremeta_centena_milhar:
429.    LDI $9
430.    STA @4 # Limpando endereço de dezena de milhar
431.    STA @511 # Limpando Key 0
432.    LDA @5 # Carregar o acumulador com o endereço da centena
433.    CEQ @50 # Compara o valor do acumulador (centena) com o valor 0
434.    JEQ %zerou # Se for igual a 0, vai para o atualiza_displays
435.    SUB @51 # Subtrai 1 no acumulador
436.    STA @5 # Armazena o valor do acumulador no endereço das centenas
437.    JMP %atualiza_displays # Se não for igual, volta para o label atualiza_displays
438.
439.    zerou:
440.    LDI $0
441.    STA @5 # Limpando endereço de centena de milhar
442.    STA @4 # Limpando endereço de dezena de milhar
443.    STA @3 # Limpando endereço de milhar
```

```
444.    STA @2 # Limpando endereço de centena
445.    STA @1 # Limpando endereço de dezena
446.    STA @0 # Limpando endereço de unidade
447.    STA @511 # Limpando Key 0
448.    JMP %atualiza_displays # Vai para o label atualiza_displays
449.
450.    atualiza_displays:
451.    LDA @0 # Carrega o acumulador com o endereço de unidade
452.    STA @288 # Armazena o valor do acumulador no endereço do HEX0
453.    LDA @1 # Carrega o acumulador com o endereço de dezena
454.    STA @289 # Armazena o valor do acumulador no endereço do HEX1
455.    LDA @2 # Carrega o acumulador com o endereço de centena
456.    STA @290 # Armazena o valor do acumulador no endereço do HEX2
457.    LDA @3 # Carrega o acumulador com o endereço de milhar
458.    STA @291 # Armazena o valor do acumulador no endereço do HEX3
459.    LDA @4 # Carrega o acumulador com o endereço de dezena de milhar
460.    STA @292 # Armazena o valor do acumulador no endereço do HEX4
461.    LDA @5 # Carrega o acumulador com o endereço de centena de milhar
462.    STA @293 # Armazena o valor do acumulador no endereço do HEX5
463.    JSR %verifica_centena_de_milhar # Vai para o label
    verifica_centena_de_milhar
464.    JMP %le_key # Vai para o label le_key
465.
466.    atualiza_displays_limites:
467.    LDA @10 # Carrega o acumulador com o Limites das unidades
468.    STA @288 # Armazena o valor do acumulador no endereço do HEX0
469.    LDA @11 # Carrega o acumulador com o Limites das dezenas
470.    STA @289 # Armazena o valor do acumulador no endereço do HEX1
471.    LDA @12 # Carrega o acumulador com o Limites de centena
472.    STA @290 # Armazena o valor do acumulador no endereço do HEX2
473.    LDA @13 # Carrega o acumulador com o Limites de milhar
474.    STA @291 # Armazena o valor do acumulador no endereço do HEX3
475.    LDA @14 # Carrega o acumulador com o Limites de dezena de milhar
476.    STA @292 # Armazena o valor do acumulador no endereço do HEX4
477.    LDA @15 # Carrega o acumulador com o Limites de centena de milhar
478.    STA @293 # Armazena o valor do acumulador no endereço do HEX5
479.    RET
```

```
480.
481.
482.    verifica_centena_de_milhar:
483.    LDA @5 # Carrega o acumulador com o endereço de centena de milhar
484.    CEQ @15 # Compara o valor do acumulador com o valor maximo de centena
    de milhar
485.    JEQ %verifica_dezena_de_milhar # Se for igual, vai para o label
    verifica_dezena_de_milhar
486.    RET
487.
488.    verifica_dezena_de_milhar:
489.    LDA @4 # Carrega o acumulador com o endereço de dezena de milhar
490.    CEQ @14 # Compara o valor do acumulador com o valor maximo de dezena de
    milhar
491.    JEQ %verifica_milhar # Se for igual, vai para o label verifica_milhar
492.    RET
493.
494.    verifica_milhar:
495.    LDA @3 # Carrega o acumulador com o endereço de milhar
496.    CEQ @13 # Compara o valor do acumulador com o valor maximo de milhar
497.    JEQ %verifica_centena # Se for igual, vai para o label verifica_centena
498.    RET
499.
500.    verifica_centena:
501.    LDA @2 # Carrega o acumulador com o endereço de centena
502.    CEQ @12 # Compara o valor do acumulador com o valor maximo de centena
503.    JEQ %verifica_dezena # Se for igual, vai para o label verifica_dezena
504.    RET
505.
506.    verifica_dezena:
507.    LDA @1 # Carrega o acumulador com o endereço de dezena
508.    CEQ @11 # Compara o valor do acumulador com o valor maximo de dezena
509.    JEQ %verifica_unidade # Se for igual, vai para o label verifica_unidade
510.    RET
511.
512.    verifica_unidade:
513.    LDA @0 # Carrega o acumulador com o endereço de unidade
```

```
514.  CEQ @10 # Compara o valor do acumulador com o valor maximo de unidade
515.  JEQ %final # Se for igual, vai para o label final
516.  RET
517.
518.  RESET_FPGA:
519.  STA @511 # Limpando key 0
520.  STA @510 # Limpando key 1
521.  STA @509 # Limpando reset_key
522.  LDI $0 # Carregando 0 no acumulador
523.  STA @0 # Limpando endereço de unidade
524.  STA @1 # Limpando endereço de dezena
525.  STA @2 # Limpando endereço de centena
526.  STA @3 # Limpando endereço de milhar
527.  STA @4 # Limpando endereço de dezena de milhar
528.  STA @5 # Limpando endereço de centena de milhar
529.  STA @258 # Desliga led 9
530.  STA @257 # Desliga o led 8
531.  STA @256 # Desliga o led 7 ao led 0
532.  JMP %atualiza_displays # Vai para o label atualiza_displays
533.
534.  final:
535.  LDA @51 # Carrega 1 no acumulador
536.  STA @258 # Liga led 9
537.  STA @257 # Liga o led 8
538.  LDI $255 # Carrega 255 no acumulador
539.  STA @256 # Liga o led 7 ao led 0
540.  LDA @356# Carrega o acumulador com o endereço de fpga_reset
541.  CEQ @51 # Compara o valor do acumulador com o valor 1
542.  JEQ %RESET_FPGA # Se for igual, vai para o label RESET_FPGA
543.  LDA @353 # Carrega o acumulador com o key 1
544.  CEQ @51 # Compara o valor do acumulador com o valor 1
545.  JEQ %define_limites_unidades # Se for igual, vai para o label incrementa
546.  JMP %final
547.
548.
```

REFERÊNCIAS

1. Assembler : https://github.com/Insper/DesignComputadores/tree/master/AssemblerASM_BIN_VHDL de Marco - Mello, adaptado e alterado