Design de Computadores

Entrega Final – Relógio

Alunos: João Victor Pazotti Silva e Bruno Saboya

Introdução

O objetivo dessa entrega é criar um reloógio com as seguintes funcionalidades: (i) incrementar valores automaticamente igual um relógio; (ii) ter um sistema para poder *setar* um valor determinado como se fosse arrumar as horas e começar a contar a partir do horário definido; e (iii) ter displays visuais para mostrar a contagem e os estados por meio de LED's. Neste relatório, será descrita a implementação do processador, detalhando como foram atendidos todos os requisitosdo projeto, tal como suas descrições.

Arquitetura do processador

Diferentemente da entrega intermediária do contador onde a arquitetura do processador era de Acumulador, nessa entrega era obrigatório alterar para Registrador-Memória.

Total de instruções e sua sintaxe

Temos um total de 13 instruções, sendo 11 delas as que já utilizávamos durante as aulas, e 2 delas foram criadas por nós alunos. São elas:

- NOP
    - Significa: Sem Operação;
    - Tipo de argumento: Nenhum argumento, a instrução de memória é composta pelo número 0 em binário;
    - Mnemônico: 0000;
- LDA
    - Significa: Carrega o acumulador A com o valor da memória;
    - Tipo de argumento: Valor que deseja ler armazenado no endereço de memória;
    - Mnemônico: 0001
- SOMA
    - Significa: Somar o valor que se encontra no acumulador A com a entrada B da ULA, originária da memória, e armazena o resultado no acumulador A;

    - Tipo de argumento: Endereço da memória com o valor que deseja somar com o valor que esteja armazenado no acumulador A;
    - Mnemônico: 0010
- SUB
    - Significa: Subtrai o valor que se encontra no acumulador A com a entrada B da ULA, originária da memória, e armazena o resultado de volta no acumulador A;
    - Tipo de argumento: Endereço da memória com o valor que deseja subtrair com o valor armazenado no acumulador A;
    - Mnemônico: 0011
- LDI
    - Significa: Carrega o valor imediato para o acumulador A;
    - Tipo de argumento: Valor que deseja carregar no acumulador;
    - Mnemônico: 0100
- STA
    - Significa: Salva o valor do acumulador A para a memória
    - Tipo de argumento: Valor do acumulador que deseja salvar na memória
    - Mnemônico: 0101
- JMP
    - Significa: Desvio de execução
    - Tipo de argumento: Posição de memória que se deseja desviar
    - Mnemônico: 0110
- JEQ
    - Significa: Desvio de execução com uma condicional de igualdade para ser cumprida
    - Tipo de argumento: Posição de memória de instruções que se deseja desviar em caso da condição de igualdade seja verdadeira;
    - Mnemônico: 0111
- CEQ
    - Significa: Comparação de igualdade
    - Tipo de argumento: Endereço da memória que possui um valor e tenha a intenção de se comparar com o valor presente no acumulador A. Caso esses dois valores sejam iguais, uma flag é levantada e recebe valor igual a 1;
    - Mnemônico: 1000
- JRS
    - Significa: Desvio de execução para uma sub-rotina
    - Tipo de argumento: Posição da memória de instruções que possui o começo da sub-rotina
    - Mnemônico: 1001
- RET
    - Significa: Retorno da execução da sub-rotina
    - Tipo de argumento: Não possui argumento, apenas sai da sub-rotina de volta para as instruções APÓS a chamada e a execução da sub-rotina

- Mnemônico: 1010

- GT
  - Significa: Compara o valor do acumulador com o valor da memória e verifica se o acumulador é maior que o valor no acumulador
  - Tipo de argumento: Posição da memória que possui um valor que se deseja comparar com o acumulador;
  - Mnemônico: 1011

- JGT
  - Significa: Desvio de execução para uma sub-rotina caso uma comparação seja verdadeira
  - Tipo de argumento: Posição da memória de instruções que possui o começo da sub-rotina após a comparação anterior
  - Mnemônico: 1100

Formato das Instruções

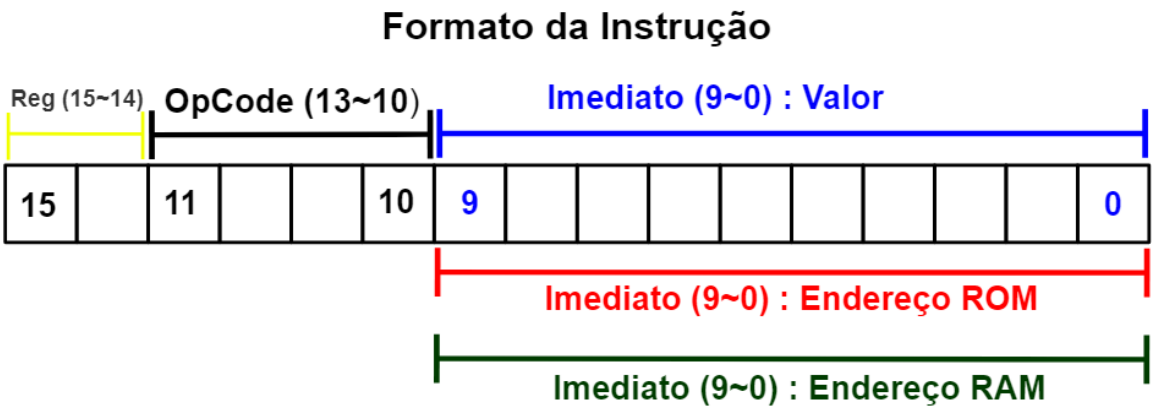

Imagem retirada da Aula 6 produzida pelo professor Paulo Santos – Alterada para suprir nossas mudanças

Para o projeto do relógio, utilizamos um formato de instrução parecido com o que utilizamosao longo das aulas. O número total de bits do vetor da instrução possui 16 bits, com os 2 mais significativos representando os Registradores das instruções, os bits de 13 à 10 estão reservados para o Opcode utilizado pela instrução e o resto do vetor, 9 bits do imediato são reservados para o destino do JMP (Jump), Endereço RAM ou então um valor a ser utilizado em alguma operação definida previamente.

Fluxo de dados para o processador

Como pode ser visto na imagem abaixo na qual ilustra a nossa CPU junto dos componentes que a compõe e para que o nosso projeto seja viável foi necessário realizar algumas adições como por exemplo as instruções GT e JGT. Com a criação dessas duas novas instruções cria-se a necessidade ne uma nova saída da ULA que se armazena uma Flag > em um Flip-flop em que caso a comparação das entradas A com a B da ULA o Flip-

flop armazena um valor binário igual a 1 e para que esse Flip-Flop funcione é necessário acrescentar também um novo ponto de controle chamado HabFlagJGT que foi conectado a Lógica de Desvio.

Outro ponto de controle implementado foi o JGT que verifica justamente a Flag> e realiza o desvio.

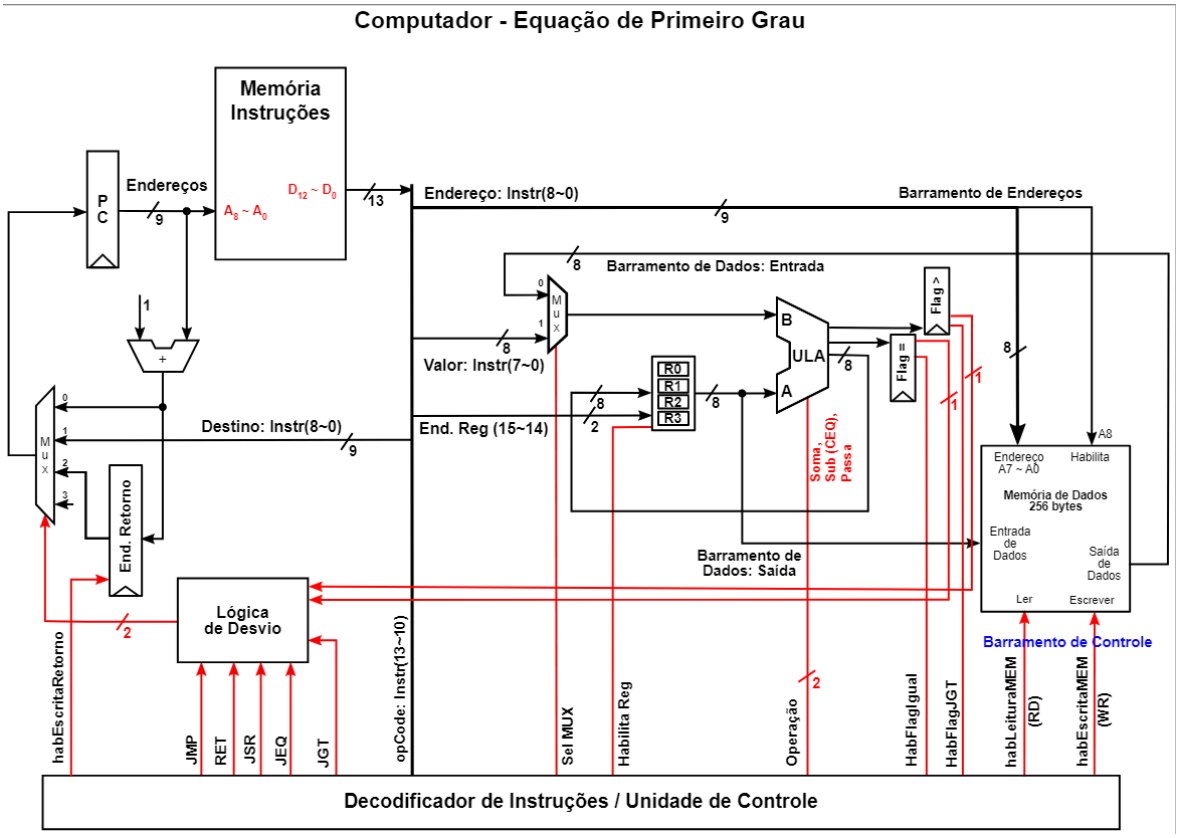

Imagem retirada da Aula 7 do Professor Paulo Santos e editada para contemplar as mudanças

Pontos de Controle e Utilizações

Mnemônico	Código Binário	JGT	Hab Flag JGT	Hab EscritaRet	JMP	RET	JSR	JEQ	SelMUX	Hab_A	Operação	Hab Flag =	RD	WR
NOP	0000	0	0	0	0	0	0	0	X	0	xx	0	0	0
LDA	0001	0	0	0	0	0	0	0	0	1	10	1	1	0
SOMA	0010	0	0	0	0	0	0	0	0	1	01	1	1	0
SUB	0011	0	0	0	0	0	0	0	0	1	00	1	1	0
LDI	0100	0	0	0	0	0	0	0	1	1	10	1	0	0
STA	0101	0	0	0	0	0	0	0	0	0	xx	0	0	1
JMP	0110	0	0	0	1	0	0	0	X	0	xx	0	0	0
JEQ	0111	0	0	0	0	0	0	1	X	0	xx	0	0	0

CEQ	1000	0	0	0	0	0	0	0	0	0	00	1	1	0
JSR	1001	0	0	1	0	0	1	0	0	0	xx	0	0	0
RET	1010	0	0	0	0	1	0	0	0	0	xx	0	0	0
GT	1011	0	1	0	0	0	0	0	0	1	0	0	0	0
JGT	1100	1	0	0	0	0	0	0	0	0	0	0	0	0

Diagrama de Conexão do Processador

A seguir um diagrama no qual mostra as conexões existentes dos periféricos que compõem a nossa CPU. Visto que aprendemos essa CPU em sala de aula, não houve necessidade alguma de realizar algum tipo de mudança das conexões e dos periféricos em si para a entrega final do Relógio.

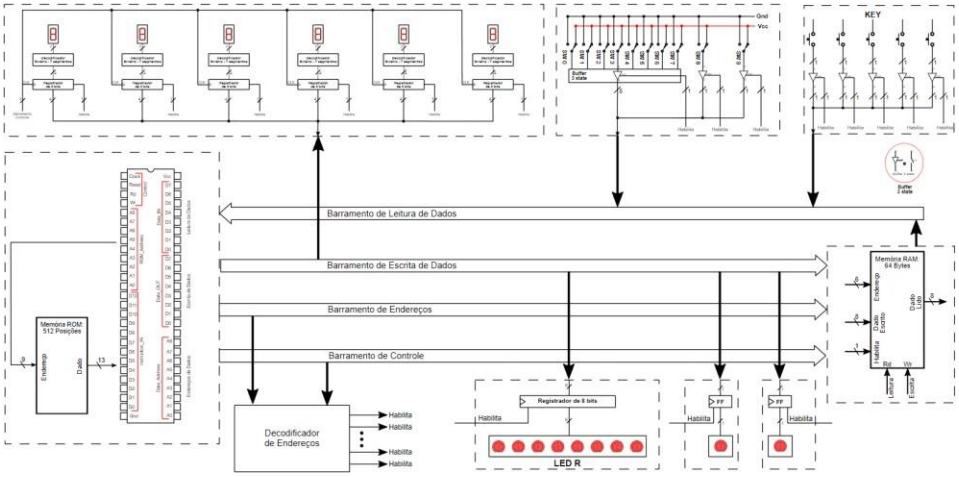

Imagem retirada da Aula 7 do Professor Paulo Santos

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 ~ 507	Reservado	–	–	7
508	Limpa Leitura KEY2	-	-	7
509	Limpa Leitura FPGA_Reset	-	Escrita	7
510	Limpa Leitura KEY1	–	Escrita	7
511	Limpa Leitura KEY0	–	Escrita	7

Utilização da Placa com o projeto

Nesse ponto do relatório serão abordados as ferramentas e os conhecimentos necessários para se utilizar o projeto do relógio na placa FPGA.

Assim que o projeto é compilado e implementado na placa,todos os LED's estarão apagados assim como os HEX's possuirão valor igual a zero e o relógio inicia sua contagem progressiva.
Como de praxe, deixe todas as Chaves HH na posição para baixo. Após esse procedimento será possível utilizar de três funcionalidades, sendo elas:

- **Botão FPGA_Reset** -> Possui intuito de resetar todos os LEDS da placa. Só é aplicável o uso do botão quando quiser "desligar" o alarme, qualquer outro uso é contraintuitivo ao uso do relógio;
- **Botão KEY_1** -> Entra no modo de definição do horário específico. Ao apertar o botão KEY1 pela primeira vez, nos HEX's aparecerá os todos os valores zerados e um LED logo abaixo do HEX específico ficará asceso, indicando em qual HEX será definido o valor. Para alterar o valor a ser mostrado basta utilizar-se das chaves HH SW_3 até SW_0, lembrando que o valor colocado está em binário. Para passar para a próxima casa de definição, ou seja, começamos na casa das unidades dos segundos e para irmos para a definição da cada das dezenas dos segundos, basta apertar novamente o botão KEY1. Dessa vez, como em um relógio convencional os segundos totais não passam dos 59s, mesmo se tentar colocar um valor binário maior que 5, será mostrado o valor limite que é 5 e esse valor fica registrado caso seja colocado um valor menor que 5, esse valor fica registrado. Apertando o botão KEY1 passa-se para as outras casas a ser definidas e quando todas as casas forem definidas pelo usuário, o relógio volta a contar a partir da hora definida pela etapa.
    - **IMPORTANTE**: Quando chegar na casa das horas, ou seja, alterar o valor do HEX 4, caso um valor maior que 3 seja selecionado, ao apertar o KEY1 e ir para a casa das dezenas das horas, não será permitido escolher um número maior que 1. Caso contrário será permitido colocar um valor igual a 2;
    - **OBSERVAÇÃO 1**: Ao entrar no modo de escolha do limite, nenhum outro botão (fora o KEY_1) assim como nenhuma outra chave (fora as chaves SW0 até SW7) irão funcionar. Para que outros botões voltem a funcionar é necessário sair do modo de escolha do limite e isso será visivel quando nenhum LED ficar asceso;
    - **OBSERVAÇÃO 2:** É permitido escolher outro horário de início depois de ter escolhido um anteriormente, basta repetir os passos descritos acima.
- **Botão KEY_2** -> Ao apertar esse botão pela primeira vez, o LED 6 ficará aceso indicando que o alarme foi ativado. A princípio, apertar o botão não acontece nada tirando acender o LED 6, para usar sua funcionalidade de alarme basta seguir o mesmo procedimento do KEY_1 no qual defini-se a hora do relógio. Porém, diferentemente dessa outra etapa, o relógio continua contando do momento do qual foi interrompido, e quando o horário do relógio for igual ao do configurado do o LED 9 ficará aceso indicando que o alarme foi disparado;
- **CHAVE SW9**-> Essa chave HH possui como função alterar a base de tempo da contagem do relógio. Como já mencionado na parte de utilização inicial da placa, recomenda-se iniciar o projeto com todas as chaves destivadas (posição para baixo), nesse modo o relógio conta de 1 em 1 segundo, caso ela esteja ativa (posição para cima) o relógio passa a contar 30 minutos em 1 segundo ou seja, um ciclo (24h) em 48 segundos.

Programa em Assembly

```
tmp(0) := R0 & NOP  & '0' & x"00";      -- NOP
tmp(1) := R0 & LDI  & '0' & x"00";      -- LDI R0, $0      #Início do Setup
tmp(2) := R0 & STA  & '1' & x"20";      -- STA R0, @288#Zerando hexas
tmp(3) := R0 & STA  & '1' & x"21";      -- STA R0, @289
tmp(4) := R0 & STA  & '1' & x"22";      -- STA R0, @290
tmp(5) := R0 & STA  & '1' & x"23";      -- STA R0, @291
tmp(6) := R0 & STA  & '1' & x"24";      -- STA R0, @292
tmp(7) := R0 & STA  & '1' & x"25";      -- STA R0, @293
tmp(8) := R0 & STA  & '1' & x"00";      -- STA R0, @256#Zerando leds
tmp(9) := R0 & STA  & '1' & x"01";      -- STA R0, @257
tmp(10) := R0 & STA  & '1' & x"02";     -- STA R0, @258
tmp(11) := R0 & STA  & '0' & x"00";     -- STA R0, @0    #Armazenando 0 em unidade, dezena,
centena, etc
tmp(12) := R0 & STA  & '0' & x"01";     -- STA R0, @1
tmp(13) := R0 & STA  & '0' & x"02";     -- STA R0, @2
tmp(14) := R0 & STA  & '0' & x"03";     -- STA R0, @3
tmp(15) := R0 & STA  & '0' & x"04";     -- STA R0, @4
tmp(16) := R0 & STA  & '0' & x"05";     -- STA R0, @5
tmp(17) := R0 & STA  & '0' & x"06";     -- STA R0, @6    #Constante de comparacao (0)
tmp(18) := R0 & STA  & '1' & x"FE";     -- STA R0, @510
tmp(19) := R0 & STA  & '1' & x"FF";     -- STA R0, @511
tmp(20) := R0 & STA  & '1' & x"FD";     -- STA R0, @509
tmp(21) := R0 & STA  & '1' & x"FC";     -- STA R0, @508
tmp(22) := R0 & LDI  & '0' & x"01";     -- LDI R0, $1
tmp(23) := R0 & STA  & '0' & x"07";     -- STA R0, @7    #Constante de Incremento (1)
tmp(24) := R0 & LDI  & '0' & x"0A";     -- LDI R0, $10
tmp(25) := R0 & STA  & '0' & x"08";     -- STA R0, @8    #Constante de limite no display (10)
tmp(26) := R0 & LDI  & '0' & x"00";     -- LDI R0, $0
tmp(27) := R0 & STA  & '0' & x"09";     -- STA R0, @9    #Limite de contagem em unidade,
dezena, centena, etc
tmp(28) := R0 & STA  & '0' & x"0A";     -- STA R0, @10
tmp(29) := R0 & STA  & '0' & x"0B";     -- STA R0, @11
tmp(30) := R0 & STA  & '0' & x"0C";     -- STA R0, @12
tmp(31) := R0 & STA  & '0' & x"0D";     -- STA R0, @13
tmp(32) := R0 & STA  & '0' & x"0E";     -- STA R0, @14
tmp(33) := R0 & STA  & '0' & x"0F";     -- STA R0, @15
tmp(34) := R0 & LDI  & '0' & x"09";     -- LDI R0, $9
tmp(35) := R0 & STA  & '0' & x"10";     -- STA R0, @16  #Constante de limite de valor(9)
tmp(36) := R0 & LDI  & '0' & x"05";     -- LDI R0, $5
tmp(37) := R0 & STA  & '0' & x"11";     -- STA R0, @17  #Constante de limite de valor de minutos
e segundos(5)
tmp(38) := R0 & LDI  & '0' & x"03";     -- LDI R0, $3
tmp(39) := R0 & STA  & '0' & x"12";     -- STA R0, @18  #Constante de limite de valor de horas1
(3)
tmp(40) := R0 & LDI  & '0' & x"02";     -- LDI R0, $2
tmp(41) := R0 & STA  & '0' & x"13";     -- STA R0, @19  #Constante de limite de valor de horas2
(2)
tmp(42) := R0 & LDI  & '0' & x"06";     -- LDI R0, $6
tmp(43) := R0 & STA  & '0' & x"14";     -- STA R0, @20  #Constante de limite de valor (6)
tmp(44) := R0 & LDI  & '0' & x"04";     -- LDI R0, $4
tmp(45) := R0 & STA  & '0' & x"15";     -- STA R0, @21  #Constante de limite de valor (4)
tmp(46) := R0 & LDI  & '0' & x"00";     -- LDI R0, $0
tmp(47) := R0 & STA  & '0' & x"16";     -- STA R0, @22  #Flag(0)
```

```
tmp(48) := R0 & STA  & '0' & x"1E";      -- STA R0, @30  # Armazena valor para operacao
tmp(49) := R0 & STA  & '0' & x"1F";      -- STA R0, @31
tmp(50) := R0 & STA  & '0' & x"20";      -- STA R0, @32
tmp(51) := R0 & STA  & '0' & x"21";      -- STA R0, @33
tmp(52) := R0 & STA  & '0' & x"22";      -- STA R0, @34
tmp(53) := R0 & STA  & '0' & x"23";      -- STA R0, @35
tmp(54) := R0 & NOP  & '0' & x"00";      -- NOP   #Loop principal
tmp(55) := R2 & LDA  & '1' & x"60";      -- LDA R2, @352          # Le o valor de KEY0
tmp(56) := R2 & CEQ  & '0' & x"06";      -- CEQ R2, @6    # Compara o valor de KEY0 com 0
tmp(57) := R0 & JEQ  & '0' & x"9F";      -- JEQ @PULA1  # Se for igual a 0, nao incrementa e
atualiza os displays
tmp(58) := R2 & STA  & '1' & x"FF";      -- STA R2, @511#Limpa a leitura de KEY0
tmp(59) := R2 & LDA  & '0' & x"00";      -- LDA R2, @0    #Carrega o valor da unidade no
acumulador
tmp(60) := R2 & SOMA  & '0' & x"07";  -- SOMA R2, @7           #Incrementa 1 na unidade
tmp(61) := R2 & CEQ  & '0' & x"08";      -- CEQ R2, @8    #Compara unidade com 10
tmp(62) := R0 & JEQ  & '0' & x"66";      -- JEQ @UNIDADEPASSOU          #Se for igual a 10,
incrementa a dezena
tmp(63) := R2 & STA  & '0' & x"00";      -- STA R2, @0    #Se for diferente de 10, armazena o valor
da unidade
tmp(64) := R2 & LDA  & '1' & x"61";      -- LDA R2, @353          # Le o valor de KEY1
tmp(65) := R2 & CEQ  & '0' & x"06";      -- CEQ R2, @6    # Compara o valor de KEY1 com 0
tmp(66) := R0 & JEQ  & '0' & x"44";      -- JEQ @CONFEREKEY2  # Se for igual a 0, ignora a
subrotina de configuracao de hora
tmp(67) := R0 & JSR  & '0' & x"AD";      -- JSR @CONFIGHORA    # Se for diferente de 0, entra na
sub rotina de configuracao de hora
tmp(68) := R0 & NOP  & '0' & x"00";      -- NOP
tmp(69) := R0 & JSR  & '1' & x"8A";      -- JSR @CHECALIMITE    # Verifica se os valores estao
dentro dos limites
tmp(70) := R2 & LDA  & '1' & x"62";      -- LDA R2, @354          # Le o valor de KEY2
tmp(71) := R2 & CEQ  & '0' & x"06";      -- CEQ R2, @6    # Compara o valor de KEY2 com 0
tmp(72) := R0 & JEQ  & '0' & x"5C";      -- JEQ @FPGA_RESET     # Se for igual a 0, ignora a flag de
hora
tmp(73) := R2 & STA  & '1' & x"FC";      -- STA R2, @508#Limpa a leitura de KEY2
tmp(74) := R2 & LDA  & '0' & x"06";      -- LDA R2, @6
tmp(75) := R2 & STA  & '1' & x"02";      -- STA R2, @258
tmp(76) := R2 & LDI  & '0' & x"01";      -- LDI R2, $1       # Se for diferente de 0, armazena 1 no
acumulador
tmp(77) := R2 & STA  & '0' & x"16";      -- STA R2, @22  #Armazena 1 na flag
tmp(78) := R2 & LDI  & '0' & x"40";      -- LDI R2, $64
tmp(79) := R2 & STA  & '1' & x"00";      -- STA R2, @256
tmp(80) := R2 & LDA  & '0' & x"00";      -- LDA R2, @0
tmp(81) := R2 & STA  & '0' & x"1E";      -- STA R2, @30
tmp(82) := R2 & LDA  & '0' & x"01";      -- LDA R2, @1
tmp(83) := R2 & STA  & '0' & x"1F";      -- STA R2, @31
tmp(84) := R2 & LDA  & '0' & x"02";      -- LDA R2, @2
tmp(85) := R2 & STA  & '0' & x"20";      -- STA R2, @32
tmp(86) := R2 & LDA  & '0' & x"03";      -- LDA R2, @3
tmp(87) := R2 & STA  & '0' & x"21";      -- STA R2, @33
tmp(88) := R2 & LDA  & '0' & x"04";      -- LDA R2, @4
tmp(89) := R2 & STA  & '0' & x"22";      -- STA R2, @34
tmp(90) := R2 & LDA  & '0' & x"05";      -- LDA R2, @5
tmp(91) := R2 & STA  & '0' & x"23";      -- STA R2, @35
tmp(92) := R0 & NOP  & '0' & x"00";      -- NOP
tmp(93) := R2 & LDA  & '1' & x"64";      -- LDA R2, @356          # Le o valor de FPGA_RESET
tmp(94) := R2 & CEQ  & '0' & x"06";      -- CEQ R2, @6    # Compara o valor de FPGA_RESET com 0
```

```
tmp(95) := R0 & JEQ  & '0' & x"36";      -- JEQ @INICIOLOOP
tmp(96) := R2 & STA  & '1' & x"FD";      -- STA R2, @509#Limpa a leitura de FPGA_RESET
tmp(97) := R2 & LDA  & '0' & x"06";      -- LDA R2, @6
tmp(98) := R2 & STA  & '1' & x"02";      -- STA R2, @258
tmp(99) := R2 & STA  & '1' & x"01";      -- STA R2, @257
tmp(100) := R2 & STA  & '1' & x"00";     -- STA R2, @256
tmp(101) := R0 & JMP  & '0' & x"36";     -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(102) := R0 & NOP  & '0' & x"00";     -- NOP
tmp(103) := R2 & LDA  & '0' & x"06";     -- LDA R2, @6    #Carrega 0 no acumulador
tmp(104) := R2 & STA  & '0' & x"00";     -- STA R2, @0    #Zera a unidade
tmp(105) := R2 & LDA  & '0' & x"01";     -- LDA R2, @1    #Carrega o valor da dezena no
acumulador
tmp(106) := R2 & SOMA  & '0' & x"07";            -- SOMA R2, @7           #Incrementa 1 na dezena
tmp(107) := R2 & CEQ  & '0' & x"14";     -- CEQ R2, @20 #Compara dezena com 6
tmp(108) := R2 & JEQ  & '0' & x"6F";     -- JEQ R2, @DEZENAPASSOU      #Se for igual a 6,
incrementa a centena
tmp(109) := R2 & STA  & '0' & x"01";     -- STA R2, @1    #Se for diferente de 6, armazena o valor
da dezena
tmp(110) := R0 & JMP  & '0' & x"36";     -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(111) := R0 & NOP  & '0' & x"00";     -- NOP
tmp(112) := R2 & LDA  & '0' & x"06";     -- LDA R2, @6    #Carrega 0 no acumulador
tmp(113) := R2 & STA  & '0' & x"01";     -- STA R2, @1    #Zera a dezena
tmp(114) := R2 & LDA  & '0' & x"02";     -- LDA R2, @2    #Carrega o valor da centena no
acumulador
tmp(115) := R2 & SOMA  & '0' & x"07";            -- SOMA R2, @7           #Incrementa 1 na
centena
tmp(116) := R2 & CEQ  & '0' & x"08";     -- CEQ R2, @8    #Compara centena com 10
tmp(117) := R0 & JEQ  & '0' & x"78";     -- JEQ @CENTENAPASSOU          #Se for igual a 10,
incrementa a unidade de milhar
tmp(118) := R2 & STA  & '0' & x"02";     -- STA R2, @2    #Se for diferente de 10, armazena o valor
da centena
tmp(119) := R0 & JMP  & '0' & x"36";     -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(120) := R0 & NOP  & '0' & x"00";     -- NOP
tmp(121) := R2 & LDA  & '0' & x"06";     -- LDA R2, @6    #Carrega 0 no acumulador
tmp(122) := R2 & STA  & '0' & x"02";     -- STA R2, @2    #Zera a centena
tmp(123) := R2 & LDA  & '0' & x"03";     -- LDA R2, @3    #Carrega o valor da unidade de milhar no
acumulador
tmp(124) := R2 & SOMA  & '0' & x"07";            -- SOMA R2, @7           #Incrementa 1 na
unidade de milhar
tmp(125) := R2 & CEQ  & '0' & x"14";     -- CEQ R2, @20 #Compara unidade de milhar com 6
tmp(126) := R0 & JEQ  & '0' & x"81";     -- JEQ @UNIDADEMILHARPASSOU          #Se for igual a 6,
incrementa a dezena de milhar
tmp(127) := R2 & STA  & '0' & x"03";     -- STA R2, @3    #Se for diferente de 6, armazena o valor
da unidade de milhar
tmp(128) := R0 & JMP  & '0' & x"36";     -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(129) := R0 & NOP  & '0' & x"00";     -- NOP
tmp(130) := R2 & LDA  & '0' & x"06";     -- LDA R2, @6    #Carrega 0 no acumulador
tmp(131) := R2 & STA  & '0' & x"03";     -- STA R2, @3    #Zera a unidade de milhar
tmp(132) := R2 & LDA  & '0' & x"05";     -- LDA R2, @5    #Carrega o valor da centena de milhar no
acumulador
tmp(133) := R2 & CEQ  & '0' & x"13";     -- CEQ R2, @19 #Compara com 2
tmp(134) := R0 & JEQ  & '0' & x"8D";     -- JEQ @HORACERTADEZENAMILHAR      #Se for igual a 2,
vai pra essa condicao
tmp(135) := R2 & LDA  & '0' & x"04";     -- LDA R2, @4    #Carrega o valor da dezena de milhar no
acumulador
tmp(136) := R2 & SOMA  & '0' & x"07";            -- SOMA R2, @7           #Incrementa 1 na dezena
```

```
de milhar
tmp(137) := R2 & CEQ  & '0' & x"08";   -- CEQ R2, @8    #Compara dezena de milhar com 10
tmp(138) := R0 & JEQ  & '0' & x"94";   -- JEQ @DEZENAMILHARPASSOU           #Se for igual a 10,
incrementa a centena de milhar
tmp(139) := R2 & STA  & '0' & x"04";   -- STA R2, @4    #Se for diferente de 10, armazena o valor
da dezena de milhar
tmp(140) := R0 & JMP  & '0' & x"36";   -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(141) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(142) := R2 & LDA  & '0' & x"04";   -- LDA R2, @4    #Carrega o valor da dezena de milhar no
acumulador
tmp(143) := R2 & SOMA  & '0' & x"07";          -- SOMA R2, @7           #Incrementa 1 na dezena
de milhar
tmp(144) := R2 & CEQ  & '0' & x"15";   -- CEQ R2, @21 #Compara dezena de milhar com 4
tmp(145) := R0 & JEQ  & '0' & x"9D";   -- JEQ @VIROUAHORA   #Se for igual a 4, virou a hora
tmp(146) := R2 & STA  & '0' & x"04";   -- STA R2, @4    #Se for diferente de 4, armazena o valor
da dezena de milhar
tmp(147) := R0 & JMP  & '0' & x"36";   -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(148) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(149) := R2 & LDA  & '0' & x"06";   -- LDA R2, @6    #Carrega 0 no acumulador
tmp(150) := R2 & STA  & '0' & x"04";   -- STA R2, @4    #Zera a dezena de milhar
tmp(151) := R2 & LDA  & '0' & x"05";   -- LDA R2, @5    #Carrega o valor da centena de milhar no
acumulador
tmp(152) := R2 & SOMA  & '0' & x"07";          -- SOMA R2, @7           #Incrementa 1 na
centena de milhar
tmp(153) := R2 & CEQ  & '0' & x"12";   -- CEQ R2, @18 #Compara com 3
tmp(154) := R0 & JEQ  & '0' & x"81";   -- JEQ @UNIDADEMILHARPASSOU          #Se for igual a 3,
volta
tmp(155) := R2 & STA  & '0' & x"05";   -- STA R2, @5    #Se for diferente de 3, armazena o valor
da centena de milhar
tmp(156) := R0 & JMP  & '0' & x"36";   -- JMP @INICIOLOOP     #Retorna para o LOOP principal
tmp(157) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(158) := R0 & JMP  & '0' & x"00";   -- JMP @RESTART         #Retorna para o LOOP principal
tmp(159) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(160) := R3 & LDA  & '0' & x"00";   -- LDA R3, @0    #Le o valor das unidades
tmp(161) := R3 & STA  & '1' & x"20";   -- STA R3, @288#Armazena o valor das unidades no HEX0
tmp(162) := R3 & LDA  & '0' & x"01";   -- LDA R3, @1    #Le o valor das dezenas
tmp(163) := R3 & STA  & '1' & x"21";   -- STA R3, @289#Armazena o valor das dezenas no HEX1
tmp(164) := R3 & LDA  & '0' & x"02";   -- LDA R3, @2    #Le o valor das centenas
tmp(165) := R3 & STA  & '1' & x"22";   -- STA R3, @290#Armazena o valor das centenas no HEX2
tmp(166) := R3 & LDA  & '0' & x"03";   -- LDA R3, @3    #Le o valor das unidades de milhar
tmp(167) := R3 & STA  & '1' & x"23";   -- STA R3, @291#Armazena o valor das unidades de
milhar no HEX3
tmp(168) := R3 & LDA  & '0' & x"04";   -- LDA R3, @4    #Le o valor das dezenas de milhar
tmp(169) := R3 & STA  & '1' & x"24";   -- STA R3, @292#Armazena o valor das dezenas de milhar
no HEX4
tmp(170) := R3 & LDA  & '0' & x"05";   -- LDA R3, @5    #Le o valor das centenas de milhar
tmp(171) := R3 & STA  & '1' & x"25";   -- STA R3, @293#Armazena o valor das centenas de
milhar no HEX5
tmp(172) := R0 & JMP  & '0' & x"36";   -- JMP @INICIOLOOP     # Volta para o loop principal
tmp(173) := R0 & NOP  & '0' & x"00";   -- NOP   #Rotina de configuracao de hora
tmp(174) := R0 & LDA  & '0' & x"06";   -- LDA R0, @6    #Carrega 0 no acumulador
tmp(175) := R0 & STA  & '0' & x"00";   -- STA R0, @0    #Armazenando 0 em unidade, dezena,
centena, etc
tmp(176) := R0 & STA  & '0' & x"01";   -- STA R0, @1
tmp(177) := R0 & STA  & '0' & x"02";   -- STA R0, @2
tmp(178) := R0 & STA  & '0' & x"03";   -- STA R0, @3
```

```
tmp(179) := R0 & STA  & '0' & x"04";    -- STA R0, @4
tmp(180) := R0 & STA  & '0' & x"05";    -- STA R0, @5
tmp(181) := R0 & STA  & '1' & x"20";    -- STA R0, @288# Zera o HEX1
tmp(182) := R0 & STA  & '1' & x"21";    -- STA R0, @289# Zera o HEX2
tmp(183) := R0 & STA  & '1' & x"22";    -- STA R0, @290# Zera o HEX3
tmp(184) := R0 & STA  & '1' & x"23";    -- STA R0, @291# Zera o HEX4
tmp(185) := R0 & STA  & '1' & x"24";    -- STA R0, @292# Zera o HEX5
tmp(186) := R0 & STA  & '1' & x"25";    -- STA R0, @293# Zera o HEX6
tmp(187) := R1 & LDI  & '0' & x"01";    -- LDI R1, $1       # Carrega o valor 1
tmp(188) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota no endereco dos LEDS(7-0)
tmp(189) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Limpa a leitura de KEY1
tmp(190) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(191) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(192) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(193) := R0 & JGT  & '0' & x"C3";    -- JGT @VALORATUALIZADO      #Se for maior que 9,
atualiza os displays
tmp(194) := R0 & JMP  & '0' & x"C5";    -- JMP @IGNORA
tmp(195) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(196) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(197) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(198) := R1 & STA  & '1' & x"20";    -- STA R1, @288# Hex 0
tmp(199) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353           # Le KEY1
tmp(200) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara KEY1 com 0
tmp(201) := R0 & JEQ  & '0' & x"BE";    -- JEQ @ESPERAUNIDADE          #Se for 0, ou seja, nao
esta apertado, espera ate apertar
tmp(202) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Limpa a leitura de KEY1
tmp(203) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(204) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(205) := R0 & JGT  & '0' & x"CF";    -- JGT @VALORATUALIZADO2     #Se for maior que 9,
atualiza os displays
tmp(206) := R0 & JMP  & '0' & x"D1";    -- JMP @IGNORA2
tmp(207) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(208) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(209) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(210) := R1 & STA  & '0' & x"00";    -- STA R1, @0    #Armazena o valor das chaves no limite
das unidades
tmp(211) := R0 & JMP  & '0' & x"D4";    -- JMP @ESPERADEZENA
tmp(212) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(213) := R1 & LDI  & '0' & x"04";    -- LDI R1, $4       #Carrega o valor 4
tmp(214) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota o valor nos LEDS
tmp(215) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(216) := R1 & GT  & '0' & x"11";     -- GT R1, @17    #Compara com 5
tmp(217) := R0 & JGT  & '0' & x"DB";    -- JGT @VALORATUALIZADO3     #Se for maior que 5,
atualiza os displays
tmp(218) := R0 & JMP  & '0' & x"DD";    -- JMP @IGNORA3
tmp(219) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(220) := R1 & LDI  & '0' & x"05";    -- LDI R1, $5
tmp(221) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(222) := R1 & STA  & '1' & x"21";    -- STA R1, @289# Hex 1
tmp(223) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353           #Le o valor de KEY1 novamente
tmp(224) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(225) := R0 & JEQ  & '0' & x"D4";    -- JEQ @ESPERADEZENA#Se for igual a 0, ficar em LOOP
"esperando" o valor mudar
tmp(226) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(227) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
```

```
tmp(228) := R1 & GT  & '0' & x"11";     -- GT R1, @17    #Compara com 5
tmp(229) := R0 & JGT  & '0' & x"E7";    -- JGT @VALORATUALIZADO4     #Se for maior que 5,
atualiza os displays
tmp(230) := R0 & JMP  & '0' & x"E9";    -- JMP @IGNORA4
tmp(231) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(232) := R1 & LDI  & '0' & x"05";    -- LDI R1, $5
tmp(233) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(234) := R1 & STA  & '0' & x"01";    -- STA R1, @1    #Armazena o valor das chaves no limte
das dezenas
tmp(235) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(236) := R1 & LDI  & '0' & x"10";    -- LDI R1, $16    # Carrega o valor 16 no acumulador
tmp(237) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota o valor nos LEDS
tmp(238) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(239) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(240) := R0 & JGT  & '0' & x"F2";    -- JGT @VALORATUALIZADO5     #Se for maior que 9,
atualiza os displays
tmp(241) := R0 & JMP  & '0' & x"F4";    -- JMP @IGNORA5
tmp(242) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(243) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(244) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(245) := R1 & STA  & '1' & x"22";    -- STA R1, @290# Hex 2
tmp(246) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353           #Le o valor de KEY1 novamente
tmp(247) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(248) := R0 & JEQ  & '0' & x"EB";    -- JEQ @ESPERACENTENA          #Se for igual a 0, ficar em
LOOP "esperando" o valor mudar
tmp(249) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(250) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(251) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(252) := R0 & JGT  & '0' & x"FE";    -- JGT @VALORATUALIZADO6     #Se for maior que 9,
atualiza os displays
tmp(253) := R0 & JMP  & '1' & x"00";    -- JMP @IGNORA6
tmp(254) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(255) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(256) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(257) := R1 & STA  & '0' & x"02";    -- STA R1, @2    #Armazena o valor das chaves no limite
das centenas
tmp(258) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(259) := R1 & LDI  & '0' & x"20";    -- LDI R1, $32    # Carrega o valor 32 no acumulador
tmp(260) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota o valor nos LEDS
tmp(261) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(262) := R1 & GT  & '0' & x"11";     -- GT R1, @17    #Compara com 5
tmp(263) := R0 & JGT  & '1' & x"09";    -- JGT @VALORATUALIZADO7     #Se for maior que 5,
atualiza os displays
tmp(264) := R0 & JMP  & '1' & x"0B";    -- JMP @IGNORA7
tmp(265) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(266) := R1 & LDI  & '0' & x"05";    -- LDI R1, $5
tmp(267) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(268) := R1 & STA  & '1' & x"23";    -- STA R1, @291# Hex 3
tmp(269) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353           #Le o valor de KEY1 novamente
tmp(270) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(271) := R0 & JEQ  & '1' & x"02";    -- JEQ @ESPERAUNIDADEMILHAR          #Se for igual a 0,
ficar em LOOP "esperando" o valor mudar
tmp(272) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(273) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
```

```
tmp(274) := R1 & GT  & '0' & x"11";     -- GT R1, @17    #Compara com 5
tmp(275) := R0 & JGT  & '1' & x"15";    -- JGT @VALORATUALIZADO8     #Se for maior que 5,
atualiza os displays
tmp(276) := R0 & JMP  & '1' & x"17";   -- JMP @IGNORA8
tmp(277) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(278) := R1 & LDI  & '0' & x"05";    -- LDI R1, $5
tmp(279) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(280) := R1 & STA  & '0' & x"03";    -- STA R1, @3    #Armazena o valor das chaves no limite
das unidades de milhar
tmp(281) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(282) := R1 & LDA  & '0' & x"05";   -- LDA R1, @5    #Le o valor da centena de milhar
tmp(283) := R1 & GT  & '0' & x"07";     -- GT R1, @7     #Compara com 1
tmp(284) := R0 & JGT  & '1' & x"34";    -- JGT @AJUSTADEZENAMILHAR
tmp(285) := R1 & LDI  & '0' & x"80";    -- LDI R1, $128   # Carrega o valor 128 no acumulador
tmp(286) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota o valor nos LEDS
tmp(287) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(288) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(289) := R0 & JGT  & '1' & x"23";    -- JGT @VALORATUALIZADO9     #Se for maior que 9,
atualiza os displays
tmp(290) := R0 & JMP  & '1' & x"25";   -- JMP @IGNORA9
tmp(291) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(292) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(293) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(294) := R1 & STA  & '1' & x"24";    -- STA R1, @292# Hex 4
tmp(295) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353          #Le o valor de KEY1 novamente
tmp(296) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(297) := R0 & JEQ  & '1' & x"19";    -- JEQ @ESPERADEZENAMILHAR#Se for igual a 0, ficar em
LOOP "esperando" o valor mudar
tmp(298) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(299) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(300) := R1 & GT  & '0' & x"10";     -- GT R1, @16    #Compara com 9
tmp(301) := R0 & JGT  & '1' & x"2F";    -- JGT @VALORATUALIZADO10   #Se for maior que 9,
atualiza os displays
tmp(302) := R0 & JMP  & '1' & x"31";   -- JMP @IGNORA10
tmp(303) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(304) := R1 & LDI  & '0' & x"09";    -- LDI R1, $9
tmp(305) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(306) := R1 & STA  & '0' & x"04";    -- STA R1, @4    #Armazena o valor das chaves no limite
das dezenas de milhar
tmp(307) := R0 & JMP  & '1' & x"4B";   -- JMP @ESPERACENTENAMILHAR
tmp(308) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(309) := R1 & LDI  & '0' & x"80";    -- LDI R1, $128   # Carrega o valor 128 no acumulador
tmp(310) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Bota o valor nos LEDS
tmp(311) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(312) := R1 & GT  & '0' & x"12";     -- GT R1, @18    #Compara com 3
tmp(313) := R0 & JGT  & '1' & x"3B";    -- JGT @VALORATUALIZADO13   #Se for maior que 3,
atualiza os displays
tmp(314) := R0 & JMP  & '1' & x"75";   -- JMP @IGNORA13
tmp(315) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(316) := R1 & LDI  & '0' & x"03";    -- LDI R1, $3
tmp(317) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(318) := R1 & STA  & '1' & x"24";    -- STA R1, @292# Hex 4
tmp(319) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353          #Le o valor de KEY1 novamente
tmp(320) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(321) := R0 & JEQ  & '1' & x"34";    -- JEQ @AJUSTADEZENAMILHAR#Se for igual a 0, ficar em
```

```
LOOP "esperando" o valor mudar
tmp(322) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(323) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(324) := R1 & GT  & '0' & x"12";     -- GT R1, @18    #Compara com 3
tmp(325) := R0 & JGT  & '1' & x"47";    -- JGT @VALORATUALIZADO14   #Se for maior que 3,
atualiza os displays
tmp(326) := R0 & JMP  & '1' & x"81";    -- JMP @IGNORA14
tmp(327) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(328) := R1 & LDI  & '0' & x"03";    -- LDI R1, $3
tmp(329) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(330) := R1 & STA  & '0' & x"04";    -- STA R1, @4    #Armazena o valor das chaves no limite
das dezenas de milhar
tmp(331) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(332) := R1 & LDA  & '0' & x"04";    -- LDA R1, @4    #Le o valor da dezena de milhar
tmp(333) := R1 & GT  & '0' & x"12";     -- GT R1, @18    #Compara com 3
tmp(334) := R0 & JGT  & '1' & x"6A";    -- JGT @HORACERTACENTENAMILHAR
tmp(335) := R1 & LDA  & '0' & x"06";    -- LDA R1, @6    #Carrega 0 no acumulador
tmp(336) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Zera o valor nos LEDS(7~0)
tmp(337) := R1 & LDI  & '0' & x"01";    -- LDI R1, $1       # Carrega o valor 1 no acumulador
tmp(338) := R1 & STA  & '1' & x"01";    -- STA R1, @257# Bota o valor nos LEDS
tmp(339) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(340) := R1 & GT  & '0' & x"13";     -- GT R1, @19    #Compara com 2
tmp(341) := R0 & JGT  & '1' & x"57";    -- JGT @VALORATUALIZADO11   #Se for maior que 2,
atualiza os displays
tmp(342) := R0 & JMP  & '1' & x"59";    -- JMP @IGNORA11
tmp(343) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(344) := R1 & LDI  & '0' & x"02";    -- LDI R1, $2
tmp(345) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(346) := R1 & STA  & '1' & x"25";    -- STA R1, @293# Hex 5
tmp(347) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353          #Le o valor de KEY1 novamente
tmp(348) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(349) := R0 & JEQ  & '1' & x"4B";    -- JEQ @ESPERACENTENAMILHAR          #Se for igual a 0,
ficar em LOOP "esperando" o valor mudar
tmp(350) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(351) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(352) := R1 & GT  & '0' & x"13";     -- GT R1, @19    #Compara com 2
tmp(353) := R0 & JGT  & '1' & x"63";    -- JGT @VALORATUALIZADO12   #Se for maior que 2,
atualiza os displays
tmp(354) := R0 & JMP  & '1' & x"65";    -- JMP @IGNORA12
tmp(355) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(356) := R1 & LDI  & '0' & x"02";    -- LDI R1, $2
tmp(357) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(358) := R1 & STA  & '0' & x"05";    -- STA R1, @5    #Armazena o valor das chaves no limite
das centenas de milhar
tmp(359) := R1 & LDI  & '0' & x"00";    -- LDI R1, $0
tmp(360) := R1 & STA  & '1' & x"01";    -- STA R1, @257
tmp(361) := R0 & JMP  & '1' & x"85";    -- JMP @FINALSUB
tmp(362) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(363) := R1 & LDA  & '0' & x"06";    -- LDA R1, @6    #Carrega 0 no acumulador
tmp(364) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Zera o valor nos LEDS(7~0)
tmp(365) := R1 & LDI  & '0' & x"01";    -- LDI R1, $1       # Carrega o valor 1 no acumulador
tmp(366) := R1 & STA  & '1' & x"01";    -- STA R1, @257# Bota o valor nos LEDS
tmp(367) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320          #Le o valor das chaves SW(7~0)
tmp(368) := R1 & GT  & '0' & x"07";     -- GT R1, @7      #Compara com 1
```

```
tmp(369) := R0 & JGT  & '1' & x"73";    -- JGT @VALORATUALIZADO15   #Se for maior que 1,
atualiza os displays
tmp(370) := R0 & JMP  & '1' & x"75";   -- JMP @IGNORA13
tmp(371) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(372) := R1 & LDI  & '0' & x"01";    -- LDI R1, $1
tmp(373) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(374) := R1 & STA  & '1' & x"25";    -- STA R1, @293# Hex 5
tmp(375) := R1 & LDA  & '1' & x"61";    -- LDA R1, @353           #Le o valor de KEY1 novamente
tmp(376) := R1 & CEQ  & '0' & x"06";    -- CEQ R1, @6    #Compara com 0 o valor de KEY1
tmp(377) := R0 & JEQ  & '1' & x"4B";    -- JEQ @ESPERACENTENAMILHAR          #Se for igual a 0,
ficar em LOOP "esperando" o valor mudar
tmp(378) := R1 & STA  & '1' & x"FE";    -- STA R1, @510#Se for diferente de 0, Limpa a leitura de
KEY1
tmp(379) := R1 & LDA  & '1' & x"40";    -- LDA R1, @320           #Le o valor das chaves SW(7~0)
tmp(380) := R1 & GT  & '0' & x"07";     -- GT R1, @7      #Compara com 1
tmp(381) := R0 & JGT  & '1' & x"7F";    -- JGT @VALORATUALIZADO16   #Se for maior que 1,
atualiza os displays
tmp(382) := R0 & JMP  & '1' & x"81";   -- JMP @IGNORA14
tmp(383) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(384) := R1 & LDI  & '0' & x"01";    -- LDI R1, $1
tmp(385) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(386) := R1 & STA  & '0' & x"05";    -- STA R1, @5    #Armazena o valor das chaves no limite
das centenas de milhar
tmp(387) := R1 & LDI  & '0' & x"00";    -- LDI R1, $0
tmp(388) := R1 & STA  & '1' & x"01";    -- STA R1, @257
tmp(389) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(390) := R1 & LDA  & '0' & x"16";    -- LDA R1, @22
tmp(391) := R1 & CEQ  & '0' & x"07";    -- CEQ R1, @7
tmp(392) := R0 & JEQ  & '1' & x"BC";    -- JEQ @TESTE
tmp(393) := R0 & RET  & '0' & x"00";    -- RET    #Retorna para o LOOP principal
tmp(394) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(395) := R1 & LDA  & '0' & x"00";    -- LDA R1, @0    #Le o valor das unidades
tmp(396) := R1 & CEQ  & '0' & x"0A";    -- CEQ R1, @10  #Compara com o valor limite das
unidades
tmp(397) := R0 & JEQ  & '1' & x"8F";    -- JEQ @CHECADEZENA  #Se for igual, checa se ocorre
com as dezenas
tmp(398) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(399) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(400) := R1 & LDA  & '0' & x"01";    -- LDA R1, @1    #Le o valor das dezenas
tmp(401) := R1 & CEQ  & '0' & x"0B";    -- CEQ R1, @11  #Compara com o valor limite das dezenas
tmp(402) := R0 & JEQ  & '1' & x"94";    -- JEQ @CHECACENTENA           #Se for igual, checa se
ocorre com as centenas
tmp(403) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(404) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(405) := R1 & LDA  & '0' & x"02";    -- LDA R1, @2    #Le o valor das centenas
tmp(406) := R1 & CEQ  & '0' & x"0C";    -- CEQ R1, @12  #Compara com o valor limite das
centenas
tmp(407) := R0 & JEQ  & '1' & x"99";    -- JEQ @CHECAUNIDADEMILHAR           #Se for igual,
checa se ocorre com as unidades de milhar
tmp(408) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(409) := R0 & NOP  & '0' & x"00";   -- NOP
tmp(410) := R1 & LDA  & '0' & x"03";    -- LDA R1, @3    # Le o valor das unidades de milhar
tmp(411) := R1 & CEQ  & '0' & x"0D";    -- CEQ R1, @13  # Compara com o valor limite das
unidades de milhar
tmp(412) := R0 & JEQ  & '1' & x"9E";    -- JEQ @CHECADEZENAMILHAR  #Se for igual, checa se
ocorre com as dezenas de milhar
```

```
tmp(413) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(414) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(415) := R1 & LDA  & '0' & x"04";    -- LDA R1, @4    # Le o valor das dezenas de milhar
tmp(416) := R1 & CEQ  & '0' & x"0E";    -- CEQ R1, @14  # Compara com o valor limite das
dezenas de milhar
tmp(417) := R0 & JEQ  & '1' & x"A3";    -- JEQ @CHECACENTENAMILHAR           #Se for igual,
checa se ocorre com as centenas de milhar
tmp(418) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(419) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(420) := R1 & LDA  & '0' & x"05";    -- LDA R1, @5    # Le o valor das centenas de milhar
tmp(421) := R1 & CEQ  & '0' & x"0F";    -- CEQ R1, @15  # Compara com o valor limite das
centenas de milhar
tmp(422) := R0 & JEQ  & '1' & x"A8";    -- JEQ @BATEUNOLIMITE           #Se for igual, indica que o
limite foi batido
tmp(423) := R0 & RET  & '0' & x"00";    -- RET    #Se for diferente, retorna para o LOOP principal
tmp(424) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(425) := R1 & LDA  & '0' & x"06";    -- LDA R1, @6
tmp(426) := R1 & STA  & '1' & x"00";    -- STA R1, @256# Zera o valor nos LEDS(7~0)
tmp(427) := R1 & LDA  & '0' & x"07";    -- LDA R1, @7    #Atribui o valor 1 no acumulador
tmp(428) := R1 & STA  & '1' & x"02";    -- STA R1, @258#Ativa o LED de limite atingido
tmp(429) := R0 & RET  & '0' & x"00";    -- RET    #Retorna pro LOOP principal
tmp(430) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(431) := R2 & LDA  & '0' & x"1E";    -- LDA R2, @30
tmp(432) := R2 & STA  & '0' & x"00";    -- STA R2, @0
tmp(433) := R2 & LDA  & '0' & x"1F";    -- LDA R2, @31
tmp(434) := R2 & STA  & '0' & x"01";    -- STA R2, @1
tmp(435) := R2 & LDA  & '0' & x"20";    -- LDA R2, @32
tmp(436) := R2 & STA  & '0' & x"02";    -- STA R2, @2
tmp(437) := R2 & LDA  & '0' & x"21";    -- LDA R2, @33
tmp(438) := R2 & STA  & '0' & x"03";    -- STA R2, @3
tmp(439) := R2 & LDA  & '0' & x"22";    -- LDA R2, @34
tmp(440) := R2 & STA  & '0' & x"04";    -- STA R2, @4
tmp(441) := R2 & LDA  & '0' & x"23";    -- LDA R2, @35
tmp(442) := R2 & STA  & '0' & x"05";    -- STA R2, @5
tmp(443) := R0 & JMP  & '0' & x"36";    -- JMP @INICIOLOOP
tmp(444) := R0 & NOP  & '0' & x"00";    -- NOP
tmp(445) := R1 & LDA  & '0' & x"00";    -- LDA R1, @0
tmp(446) := R1 & STA  & '0' & x"0A";    -- STA R1, @10
tmp(447) := R1 & LDA  & '0' & x"01";    -- LDA R1, @1
tmp(448) := R1 & STA  & '0' & x"0B";    -- STA R1, @11
tmp(449) := R1 & LDA  & '0' & x"02";    -- LDA R1, @2
tmp(450) := R1 & STA  & '0' & x"0C";    -- STA R1, @12
tmp(451) := R1 & LDA  & '0' & x"03";    -- LDA R1, @3
tmp(452) := R1 & STA  & '0' & x"0D";    -- STA R1, @13
tmp(453) := R1 & LDA  & '0' & x"04";    -- LDA R1, @4
tmp(454) := R1 & STA  & '0' & x"0E";    -- STA R1, @14
tmp(455) := R1 & LDA  & '0' & x"05";    -- LDA R1, @5
tmp(456) := R1 & STA  & '0' & x"0F";    -- STA R1, @15
tmp(457) := R1 & LDA  & '0' & x"06";    -- LDA R1, @6
tmp(458) := R1 & STA  & '0' & x"16";    -- STA R1, @22
tmp(459) := R0 & JMP  & '1' & x"AE";    -- JMP @RETORNO
```