Relatório do Projeto 1 - Design de Computadores

Alunos
Guilherme Carneiro Lunetta
José Rafael Martins Fernandes

ARQUITETURA DO PROCESSADOR

A arquitetura do processador utilizado no projeto é a arquitetura Registrador-Memória. Na entrega intermediária tínhamos optado por manter a arquitetura Acumulador, que havia sido utilizado majoritariamente nas aulas, contudo para essa entrega, devido a obrigatoriedade do projeto, utilizamos a arquitetura citada acima.

INSTRUÇÕES E SUA SINTAXE

No total, temos 15 instruções. São elas:

- **NOP**
  Significado: Sem Operação
  Argumento: Não tem argumento, na memória de instruções é acompanhado do valor 0 em binário
  Binário do mnemônico: 0000

- **LDA**
  Significado: Carrega o valor de um endereço da memória no registrador escolhido
  Argumento: Endereço do registrador que deseja colocar o valor salvo no endereço de memória escolhido e o endereço da memória do valor que você deseja ler, nessa ordem
  Binário do mnemônico: 0001

- **SOMA**
  Significado: Soma o valor presente no registrador escolhido com a entrada B da ULA, que vem da memória, e armazena o resultado no mesmo registrador
  Argumento: Endereço do registrador que deseja usar na soma e o endereço da memória do valor que você deseja somar com o que está armazenado no registrador, nessa ordem
  Binário do mnemônico: 0010

- **SUB**
  Significado: Subtrai o valor da entrada B da ULA, que vem da memória, do valor que está armazenado no registrador escolhido, e armazena novamente no registrador

Argumento: Endereço do registrador que deseja subtrair o valor que vem da memória e o endereço da memória do valor que você deseja subtrair do que está armazenado no registrador escolhido, nessa ordem
Binário do mnemônico: 0011

- **LDI**
  Significado: Carrega o valor do imediato no registrador escolhido
  Argumento: Endereço do registrador que deseja salvar o valor do imediato e o valor que deseja carregar no registrador, nessa ordem
  Binário do mnemônico: 0100

- **STA**
  Significado: Salva o valor que está armazenado no registrador escolhido na memória
  Argumento: Endereço do registrador com o valor que deseja guardar na memória e o endereço da memória que deseja salvar o valor, nessa ordem
  Binário do mnemônico: 0101

- **JMP**
  Significado: Desvio absoluto de execução
  Argumento: Posição da memória de instruções que deseja desviar
  Binário do mnemônico: 0110

- **JEQ**
  Significado: Desvio condicional de execução a partir de uma condição de igualdade
  Argumento: Posição da memória de instruções que deseja desviar caso a condição seja verdadeira
  Binário do mnemônico: 0111

- **CEQ**
  Significado: Comparação de igualdade entre um registrador a sua escolha e o valor armazenado em um endereço de memória, caso sejam iguais, uma flag de igualdade receberá o valor 1
  Argumento: Endereço do registrador que contém o valor a ser comparado e o endereço da memória com o valor que você deseja comparar com o valor que está presente no registrador escolhido, nessa ordem
  Binário do mnemônico: 1000

- **JSR**
  Significado: Desvio para sub rotina
  Argumento: Posição da memória de instruções que contém o início da sub rotina
  1001

- **RET**

Significado: Retorno da sub rotina
Argumento: Não tem argumento, apenas volta para a instrução da memória de instruções DEPOIS da chamada da sub rotina
Binário do mnemônico: 1010

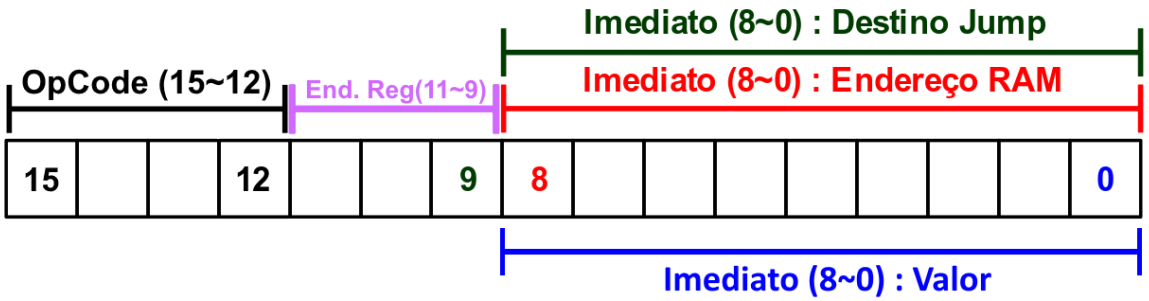
- **GRT**
  Significado: Comparação de maior que entre um registrador a sua escolha e o valor armazenado em um endereço de memória, caso seja maior, uma flag de maior que receberá o valor 1
  Argumento: Endereço do registrador que contém o valor a ser comparado e o endereço da memória com o valor que você deseja comparar com o valor que está presente no registrador escolhido, nessa ordem
  Binário do mnemônico: 1011

- **JGT**
  Significado: Desvio condicional a partir de uma condição de maior que
  Argumento: Posição da memória de instruções que deseja desviar caso a condição seja verdadeira
  Binário do mnemônico: 1100

- **ADDI**
  Significado: Soma o valor do imediato com o valor presente no registrador passado como argumento
  Argumento: O endereço do registrador e o valor que deseja colocar no imediato para ser somado, nessa ordem
  Binário do mnemônico: 1101

- **SUBI**
  Significado: Subtrai o valor presente no imediato do valor presente no registrador passado como argumento
  Argumento: O endereço do registrador e o valor que deseja colocar no imediato para ser subtraído, nessa ordem
  Binário do mnemônico: 1110

FORMATO DAS INSTRUÇÕES



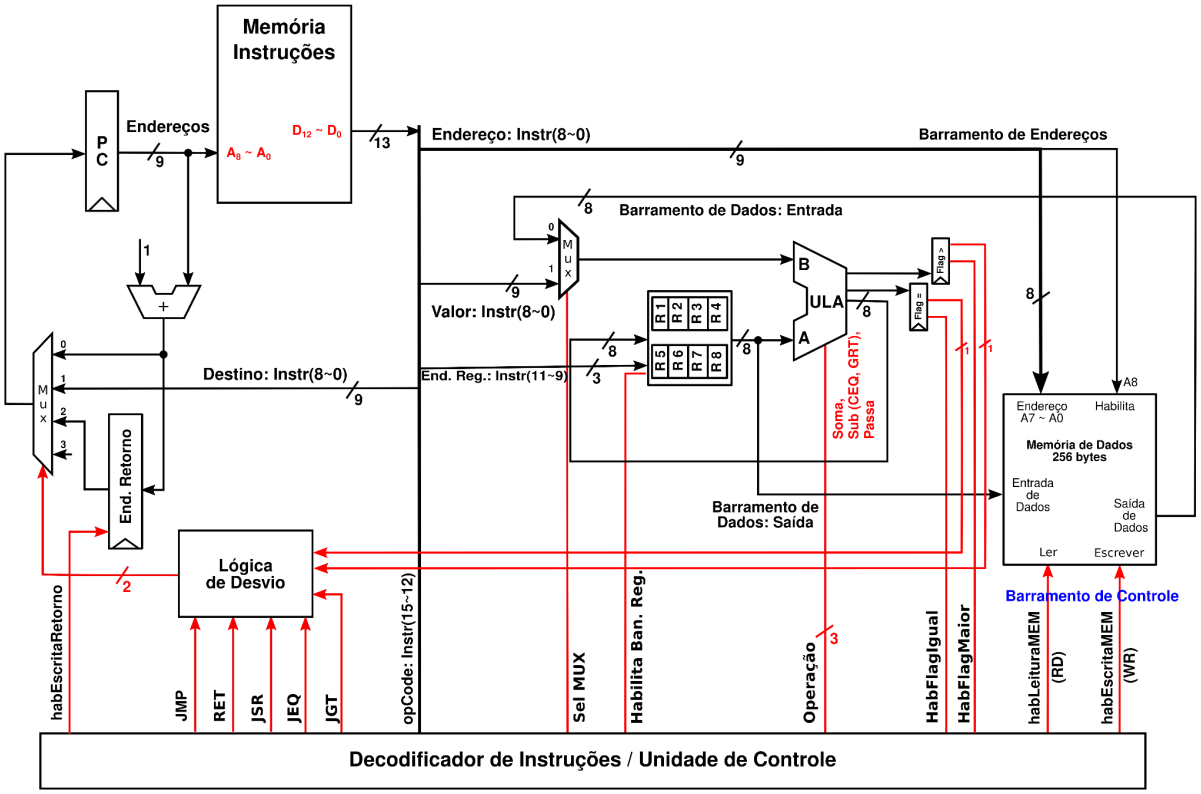
Formato da Instrução

O formato da instrução utilizado foi muito parecido com o que usamos durante as aulas, apenas com a modificação necessária para a arquitetura Registrador-Memória. Ele consiste em um vetor de 16 bits, sendo os 4 mais significativos reservados para o OpCode das instruções, os bits 11, 10 e 9 para o endereço do registrador utilizado na instrução e os 9 bits restantes do imediato para endereçar a RAM, o destino do Jump, ou um valor para ser utilizado em alguma operação. Exemplo do formato de uma instrução no Quartus:

constant LDA : std_logic_vector(3 downto 0) := "0001";
tmp(1) := LDA & *"000"* & "101100000"
Significado: LDA #0 @352, onde #0 é o endereço do registrador e @352 é o endereço da memória

FLUXO DE DADOS PARA O PROCESSADOR

A imagem abaixo ilustra a CPU e os componentes que a compõem. Para esse projeto fizemos apenas algumas modificações em relação ao que foi feito em aula, para possibilitar a implementação das instruções GRT e JGT. Dessa forma, existe uma nova saída da ULA que armazena em um Flip Flop uma Flag que indica a comparação de maior, dessa forma se a entrada A for maior que a entrada B esse Flip Flop armazena o valor binário 1. Um novo ponto de controle (Habilita Flag Maior Que) foi adicionado para habilitar esse Flip Flop, e sua saída foi ligada ao componente responsável pela Lógica de Desvio. Por fim, outro ponto de controle foi adicionado (JGT), conectado a Lógica de Desvio e responsável por avaliar a flag mencionada anteriormente e fazer o desvio ser acionado. Ainda, vale ressaltar que a imagem contém as conexões com a Memória de Instruções (ROM) e com a Memória de Dados (RAM), mas no projeto elas são componentes separados que não fazem parte da CPU e são ligadas ao processador por meio de alguns barramentos.



PONTOS DE CONTROLE E UTILIZAÇÕES

No total, nossa CPU possui 15 pontos de controle, cada um deles com sua utilização. São eles:

- **Habilita Escrita Retorno (1 bit)**
  Utilização: Utilizado para habilitar o acumulador do endereço de retorno das sub rotinas
  Usado por: JSR

- **JMP (1 bit)**
  Utilização: Utilizado para o componente lógica de desvio entender que queremos realizar um jump absoluto
  Usado por: JMP

- **RET (1 bit)**
  Utilização: Utilizado para o componente lógica de desvio entender que queremos realizar um jump para o endereço armazenado no acumulador de retorno
  Usado por: RET

- **JSR (1 bit)**

Utilização: Utilizado para o componente lógica de desvio entender que queremos realizar um jump para sub rotina
Usado por: JSR

- **JEQ (1 bit)**
  Utilização: Utilizado para o componente lógica de desvio entender que queremos realizar um jump caso a condição de igualdade seja verdadeira
  Usado por: JEQ

- **JGT (1 bit)**
  Utilização: Utilizado para o componente lógica de desvio entender que queremos realizar um jump caso a condição de maior que seja verdadeira
  Usado por: JGT

- **Seletor do MUX (1 bit)**
  Utilização: Utilizado para escolher o que vai para a entrada B da ULA, sendo as opções: valor que vem do imediato ou valor que vem da memória
  Usado por: LDI, ADDI e SUBI

- **Habilita Banco de Registradores (1 bit)**
  Utilização: Utilizado para habilitar a entrada de dados no acumulador A
  Usado por: LDA, SOMA, SUB, LDI, ADDI e SUBI

- **Operação da ULA (3 bits)**
  Utilização: Utilizado para escolher a operação que a ULA irá realizar, podendo ser: soma, subtração, passa, compara igualdade (subtração) e compara maior que (subtração)
  Usado por: Todas as instruções

- **Habilita Flag de Igualdade (1 bit)**
  Utilização: Utilizado para habilitar a entrada de dados do acumulador de igualdade
  Usado por: CEQ

- **Habilita Flag de Maior Que (1 bit)**
  Utilização: Utilizado para habilitar a entrada de dados do acumulador de maior que
  Usado por: GRT

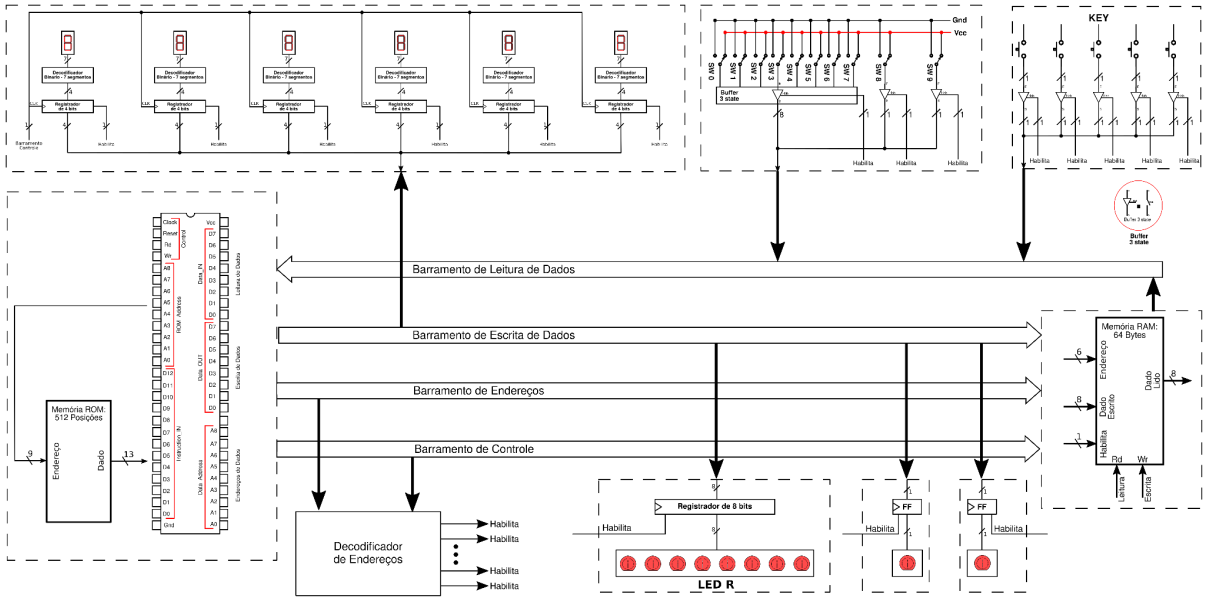
- **Read (1 bit)**
  Utilização: Utilizado para habilitar a leitura da memória RAM
  Usado por: LDA, SOMA, SUB, CEQ e GRT

- **Write (1 bit)**
  Utilização: Utilizado para habilitar a escrita na memória RAM
  Usado por: STA

Lembrando que no Quartus os pontos de controle estão exatamente nessa ordem, da esquerda para a direita.

DIAGRAMA DE CONEXÃO DO PROCESSADOR

O diagrama abaixo mostra as conexões dos periféricos com a CPU, para esse projeto não houve nenhuma modificação em relação ao que fizemos em aula.



MAPA DE REGISTRADORES

Registrador	Função	Endereço
0	Unidade de segundos do relógio	0000
1	Dezena de segundos do relógio	0001
2	Unidade de minutos do relógio	0010
3	Dezena de minutos do relógio	0011
4	Unidade de hora do relógio	0100
5	Dezena de hora do relógio	0101

6	Uso Geral	0110
7	Uso Geral	0111

MAPA DE MEMÓRIA

Nosso mapa de memória segue o que foi escolhido pelo professor nas aulas, no entanto, para o nosso projeto ter as novas funcionalidades, foi necessário endereçar novas variáveis/constantes na memória. Abaixo segue o mapa de memória completo:

Endereço em Decimal	Periférico	Largura dos Dados	Tipo de Acesso	Bloco (Página) de Memória	Nome da Variável	Variável Salva (Decimal)
0	RAM	8 bits	Leitura/ Escrita	0	Valor 0 para comparações	0
1	RAM	8 bits	Leitura/ Escrita	0	Valor 1 para comparações	1
2	RAM	8 bits	Leitura/ Escrita	0	Valor 2 para comparações	2
3	RAM	8 bits	Leitura/ Escrita	0	Valor 3 para comparações	3
4	RAM	8 bits	Leitura/ Escrita	0	-	-
5	RAM	8 bits	Leitura/ Escrita	0	-	-
6	RAM	8 bits	Leitura/ Escrita	0	-	-
7	RAM	8 bits	Leitura/ Escrita	0	-	-
8	RAM	8 bits	Leitura/ Escrita	0	-	-
9	RAM	8 bits	Leitura/ Escrita	0	Valor 9 para comparações	9
10 ~ 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 ~ 506	Reservado	–	–	7	-	-
507	Habilita Contagem 1 Segundo	–	Escrita	7		
508	Limpa Leitura 1 Segundo	–	Escrita	7		

509	Limpa Leitura KEY2	–	Escrita	7		
510	Limpa Leitura KEY1	–	Escrita	7	-	-
511	Limpa Leitura KEY0	–	Escrita	7	-	-

UTILIZAÇÃO DO PROJETO NA PLACA

Neste capítulo, falaremos como utilizar o projeto na placa FPGA. Ao rodar o projeto na placa, os HEX's estarão todos em zero, os LED's estarão todos apagados e de preferência recomendamos deixar as chaves todas abaixadas. O relógio irá começar a funcionar da forma esperada. Dito isso, você agora tem 3 funcionalidades que podem ser usadas, são elas:

- **Botão KEY1**
  Função: Entrar no modo de escolha do horário. Ao apertar o botão, o LED 0 irá acender, indicando que você está no modo de escolha do horário. Já dentro do modo de escolha, o programa vai esperar você escolher o limite da dezena de hora, unidade de hora, dezena de minuto, unidade de minuto, dezena de segundo e unidade de segundo, **UMA DE CADA VEZ**. Para escolher o valor, basta usar as chaves SW0 até a chave SW3, lembre-se que se escolhe o valor em binário. Assim que você escolher um valor, basta clicar no botão 1 novamente para confirmar sua escolha e ir para o próximo HEX. Quando confirmar sua escolha para o HEX0 (unidade de segundo) e ela for válida, você sairá automaticamente do modo de escolha. Quando sair, os HEX's irão mostrar o valor que você acabou de escolher, afinal o intuito do modo de escolha é esse.
  OBSERVAÇÃO: Quando estiver nesse modo, **NENHUM** outro botão vai funcionar, você precisa finalizar a escolha e sair do modo de escolha para que eles voltem ao normal.
  OBSERVAÇÃO 2: Não tem como você escolher um valor inválido, ou seja, você só consegue escolher de 0 até 24 horas, de 0 até 59 minutos e de 0 até 59 segundos. Se você colocar um valor inválido, o programa não vai confirmar sua escolha e não vai passar para o próximo HEX, você só vai poder continuar após colocar uma valor **VÁLIDO**.
  OBSERVAÇÃO 3: Você **PODE** escolher outro horário de início depois de ter escolhido um anteriormente, não tem problema nenhum nisso, basta repetir os passos descritos acima.

- **Chave SW9**

Função: Mudar a base de tempo do relógio. Caso a chave SW9 esteja ativada (para cima), o relógio irá funcionar em uma velocidade maior do que o normal (1 segundo). Isso foi feito por motivos de teste.
OBSERVAÇÃO: A chave desativada (para baixo) faz o relógio funcionar de 1 em 1 segundo, ou seja, normalmente e o tempo total para completar um ciclo (24 horas) é de 24 horas. A chave ativada (para cima) faz o relógio funcionar de 714 em 714 segundos, desse modo o tempo total de um ciclo (24 horas) é de 2 minutos.

- **Chave SW8**
  Função: Alternar entre contagem regressiva e contagem progressiva. Caso a chave SW8 esteja ativada (para cima), o relógio irá contar regressivamente e, caso contrário, irá contar progressivamente.
  OBSERVAÇÃO: A chave SW9 funciona da mesma forma independente da chave SW8 estar ativada ou desativada.

PROGRAMA EM ASSEMBLY

```
tmp(0) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(1) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(2) := LDI & "010" & "000000000"; -- LDI #2 $0
tmp(3) := LDI & "011" & "000000000"; -- LDI #3 $0
tmp(4) := LDI & "100" & "000000000"; -- LDI #4 $0
tmp(5) := LDI & "101" & "000000000"; -- LDI #5 $0
tmp(6) := LDI & "110" & "000000000"; -- LDI #6 $0
tmp(7) := STA & "110" & "100100000"; -- STA #6 @288
tmp(8) := STA & "110" & "100100001"; -- STA #6 @289
tmp(9) := STA & "110" & "100100010"; -- STA #6 @290
tmp(10) := STA & "110" & "100100011"; -- STA #6 @291
tmp(11) := STA & "110" & "100100100"; -- STA #6 @292
tmp(12) := STA & "110" & "100100101"; -- STA #6 @293
tmp(13) := STA & "110" & "000000000"; -- STA #6 @0
tmp(14) := LDI & "111" & "000001001"; -- LDI #7 $9
tmp(15) := STA & "111" & "000001001"; -- STA #7 @9
tmp(16) := LDI & "111" & "000000101"; -- LDI #7 $5
tmp(17) := STA & "111" & "000000101"; -- STA #7 @5
tmp(18) := LDI & "111" & "000000001"; -- LDI #7 $1
tmp(19) := STA & "111" & "000000001"; -- STA #7 @1
tmp(20) := LDI & "111" & "000000010"; -- LDI #7 $2
tmp(21) := STA & "111" & "000000010"; -- STA #7 @2
tmp(22) := LDI & "111" & "000000011"; -- LDI #7 $3
tmp(23) := STA & "111" & "000000011"; -- STA #7 @3
tmp(24) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(25) := CEQ & "111" & "000000001"; -- CEQ #7 @1
tmp(26) := JEQ & "000" & "000011100"; -- JEQ %SETINICIO
```

```
tmp(27) := JMP & "000" & "000011101"; -- JMP %PULASETINICIO
tmp(28) := JSR & "000" & "010100101"; -- JSR %SETINIT
tmp(29) := LDA & "111" & "111111011"; -- LDA #7 @507
tmp(30) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(31) := JEQ & "000" & "000011000"; -- JEQ %INICIO
tmp(32) := STA & "000" & "111111100"; -- STA #0 @508
tmp(33) := LDA & "111" & "101000001"; -- LDA #7 @321
tmp(34) := CEQ & "111" & "000000001"; -- CEQ #7 @1
tmp(35) := JEQ & "000" & "000100110"; -- JEQ %DECREMENTA
tmp(36) := JSR & "000" & "000101001"; -- JSR %RELOGIO
tmp(37) := JMP & "000" & "000011000"; -- JMP %INICIO
tmp(38) := JSR & "000" & "001110111"; -- JSR %REGRESSIVA
tmp(39) := JMP & "000" & "000011000"; -- JMP %INICIO

-- RELOGIO
tmp(41) := ADDI & "000" & "000000001"; -- ADDI #0 $1
tmp(42) := GRT & "000" & "000001001"; -- GRT #0 @9
tmp(43) := JGT & "000" & "000101110"; -- JGT %INCDEZSEC
tmp(44) := STA & "000" & "100100000"; -- STA #0 @288
tmp(45) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(46) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(47) := STA & "000" & "100100000"; -- STA #0 @288
tmp(48) := ADDI & "001" & "000000001"; -- ADDI #1 $1
tmp(49) := GRT & "001" & "000000101"; -- GRT #1 @5
tmp(50) := JGT & "000" & "000110101"; -- JGT %INCUNIMIN
tmp(51) := STA & "001" & "100100001"; -- STA #1 @289
tmp(52) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(53) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(54) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(55) := STA & "000" & "100100000"; -- STA #0 @288
tmp(56) := STA & "001" & "100100001"; -- STA #1 @289
tmp(57) := ADDI & "010" & "000000001"; -- ADDI #2 $1
tmp(58) := GRT & "010" & "000001001"; -- GRT #2 @9
tmp(59) := JGT & "000" & "000111110"; -- JGT %INCDEZMIN
tmp(60) := STA & "010" & "100100010"; -- STA #2 @290
tmp(61) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(62) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(63) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(64) := LDI & "010" & "000000000"; -- LDI #2 $0
tmp(65) := STA & "000" & "100100000"; -- STA #0 @288
tmp(66) := STA & "001" & "100100001"; -- STA #1 @289
tmp(67) := STA & "010" & "100100010"; -- STA #2 @290
tmp(68) := ADDI & "011" & "000000001"; -- ADDI #3 $1
tmp(69) := GRT & "011" & "000000101"; -- GRT #3 @5
```

```
tmp(70) := JGT & "000" & "001001001"; -- JGT %INCUNIHORA
tmp(71) := STA & "011" & "100100011"; -- STA #3 @291
tmp(72) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(73) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(74) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(75) := LDI & "010" & "000000000"; -- LDI #2 $0
tmp(76) := LDI & "011" & "000000000"; -- LDI #3 $0
tmp(77) := STA & "000" & "100100000"; -- STA #0 @288
tmp(78) := STA & "001" & "100100001"; -- STA #1 @289
tmp(79) := STA & "010" & "100100010"; -- STA #2 @290
tmp(80) := STA & "011" & "100100011"; -- STA #3 @291
tmp(81) := ADDI & "100" & "000000001"; -- ADDI #4 $1
tmp(82) := GRT & "101" & "000000001"; -- GRT #5 @1
tmp(83) := JGT & "000" & "001011000"; -- JGT %LIMITEHORA4
tmp(84) := GRT & "100" & "000001001"; -- GRT #4 @9
tmp(85) := JGT & "000" & "001011100"; -- JGT %INCDEZHORA
tmp(86) := STA & "100" & "100100100"; -- STA #4 @292
tmp(87) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(88) := GRT & "100" & "000000011"; -- GRT #4 @3
tmp(89) := JGT & "000" & "001101001"; -- JGT %ZERATUDO
tmp(90) := STA & "100" & "100100100"; -- STA #4 @292
tmp(91) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(92) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(93) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(94) := LDI & "010" & "000000000"; -- LDI #2 $0
tmp(95) := LDI & "011" & "000000000"; -- LDI #3 $0
tmp(96) := LDI & "100" & "000000000"; -- LDI #4 $0
tmp(97) := STA & "000" & "100100000"; -- STA #0 @288
tmp(98) := STA & "001" & "100100001"; -- STA #1 @289
tmp(99) := STA & "010" & "100100010"; -- STA #2 @290
tmp(100) := STA & "011" & "100100011"; -- STA #3 @291
tmp(101) := STA & "100" & "100100100"; -- STA #4 @292
tmp(102) := ADDI & "101" & "000000001"; -- ADDI #5 $1
tmp(103) := STA & "101" & "100100101"; -- STA #5 @293
tmp(104) := JMP & "000" & "001110101"; -- JMP %FIMRELOGIO
tmp(105) := LDI & "000" & "000000000"; -- LDI #0 $0
tmp(106) := LDI & "001" & "000000000"; -- LDI #1 $0
tmp(107) := LDI & "010" & "000000000"; -- LDI #2 $0
tmp(108) := LDI & "011" & "000000000"; -- LDI #3 $0
tmp(109) := LDI & "100" & "000000000"; -- LDI #4 $0
tmp(110) := LDI & "101" & "000000000"; -- LDI #5 $0
tmp(111) := STA & "000" & "100100000"; -- STA #0 @288
tmp(112) := STA & "001" & "100100001"; -- STA #1 @289
tmp(113) := STA & "010" & "100100010"; -- STA #2 @290
```

```
tmp(114) := STA & "011" & "100100011"; -- STA #3 @291
tmp(115) := STA & "100" & "100100100"; -- STA #4 @292
tmp(116) := STA & "101" & "100100101"; -- STA #5 @293
tmp(117) := RET & "000" & "000000000"; -- RET

-- REGRESSIVA
tmp(119) := CEQ & "000" & "000000000"; -- CEQ #0 @0
tmp(120) := JEQ & "000" & "001111100"; -- JEQ %DECUNISEC
tmp(121) := SUBI & "000" & "000000001"; -- SUBI #0 $1
tmp(122) := STA & "000" & "100100000"; -- STA #0 @288
tmp(123) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(124) := LDI & "000" & "000001001"; -- LDI #0 $9
tmp(125) := STA & "000" & "100100000"; -- STA #0 @288
tmp(126) := CEQ & "001" & "000000000"; -- CEQ #1 @0
tmp(127) := JEQ & "000" & "010000011"; -- JEQ %DECDEZSEC
tmp(128) := SUBI & "001" & "000000001"; -- SUBI #1 $1
tmp(129) := STA & "001" & "100100001"; -- STA #1 @289
tmp(130) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(131) := LDI & "001" & "000000101"; -- LDI #1 $5
tmp(132) := STA & "001" & "100100001"; -- STA #1 @289
tmp(133) := CEQ & "010" & "000000000"; -- CEQ #2 @0
tmp(134) := JEQ & "000" & "010001010"; -- JEQ %DECUNIMIN
tmp(135) := SUBI & "010" & "000000001"; -- SUBI #2 $1
tmp(136) := STA & "010" & "100100010"; -- STA #2 @290
tmp(137) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(138) := LDI & "010" & "000001001"; -- LDI #2 $9
tmp(139) := STA & "010" & "100100010"; -- STA #2 @290
tmp(140) := CEQ & "011" & "000000000"; -- CEQ #3 @0
tmp(141) := JEQ & "000" & "010010001"; -- JEQ %DECDEZMIN
tmp(142) := SUBI & "011" & "000000001"; -- SUBI #3 $1
tmp(143) := STA & "011" & "100100011"; -- STA #3 @291
tmp(144) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(145) := LDI & "011" & "000000101"; -- LDI #3 $5
tmp(146) := STA & "011" & "100100011"; -- STA #3 @291
tmp(147) := CEQ & "100" & "000000000"; -- CEQ #4 @0
tmp(148) := JEQ & "000" & "010011000"; -- JEQ %DECUNIHORA
tmp(149) := SUBI & "100" & "000000001"; -- SUBI #4 $1
tmp(150) := STA & "100" & "100100100"; -- STA #4 @292
tmp(151) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(152) := CEQ & "101" & "000000000"; -- CEQ #5 @0
tmp(153) := JEQ & "000" & "010011111"; -- JEQ %RESETA
tmp(154) := LDI & "100" & "000001001"; -- LDI #4 $9
tmp(155) := STA & "100" & "100100100"; -- STA #4 @292
tmp(156) := SUBI & "101" & "000000001"; -- SUBI #5 $1
```

```
tmp(157) := STA & "101" & "100100101"; -- STA #5 @293
tmp(158) := JMP & "000" & "010100011"; -- JMP %FIMREGRESSIVA
tmp(159) := LDI & "100" & "000000011"; -- LDI #4 $3
tmp(160) := LDI & "101" & "000000010"; -- LDI #5 $2
tmp(161) := STA & "100" & "100100100"; -- STA #4 @292
tmp(162) := STA & "101" & "100100101"; -- STA #5 @293
tmp(163) := RET & "000" & "000000000"; -- RET

-- SETINIT
tmp(165) := STA & "111" & "111111110"; -- STA #7 @510
tmp(166) := LDI & "111" & "000000001"; -- LDI #7 $1
tmp(167) := STA & "111" & "100000000"; -- STA #7 @256
tmp(168) := LDI & "111" & "000000000"; -- LDI #7 $0
tmp(169) := STA & "111" & "100100000"; -- STA #7 @288
tmp(170) := STA & "111" & "100100001"; -- STA #7 @289
tmp(171) := STA & "111" & "100100010"; -- STA #7 @290
tmp(172) := STA & "111" & "100100011"; -- STA #7 @291
tmp(173) := STA & "111" & "100100100"; -- STA #7 @292
tmp(174) := STA & "111" & "100100101"; -- STA #7 @293
tmp(175) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(176) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(177) := JEQ & "000" & "010101111"; -- JEQ %SETDEZHORA
tmp(178) := STA & "111" & "111111110"; -- STA #7 @510
tmp(179) := LDA & "101" & "101000000"; -- LDA #5 @320
tmp(180) := GRT & "101" & "000000010"; -- GRT #5 @2
tmp(181) := JGT & "000" & "010101111"; -- JGT %SETDEZHORA
tmp(182) := STA & "101" & "100100101"; -- STA #5 @293
tmp(183) := GRT & "101" & "000000001"; -- GRT #5 @1
tmp(184) := JGT & "000" & "011000010"; -- JGT %SETUNIHORA4
tmp(185) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(186) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(187) := JEQ & "000" & "010111001"; -- JEQ %SETUNIHORA9
tmp(188) := STA & "111" & "111111110"; -- STA #7 @510
tmp(189) := LDA & "100" & "101000000"; -- LDA #4 @320
tmp(190) := GRT & "100" & "000001001"; -- GRT #4 @9
tmp(191) := JGT & "000" & "010111001"; -- JGT %SETUNIHORA9
tmp(192) := STA & "100" & "100100100"; -- STA #4 @292
tmp(193) := JMP & "000" & "011001010"; -- JMP %SETDEZMIN
tmp(194) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(195) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(196) := JEQ & "000" & "011000010"; -- JEQ %SETUNIHORA4
tmp(197) := STA & "111" & "111111110"; -- STA #7 @510
tmp(198) := LDA & "100" & "101000000"; -- LDA #4 @320
tmp(199) := GRT & "100" & "000000011"; -- GRT #4 @3
```

```
tmp(200) := JGT & "000" & "011000010"; -- JGT %SETUNIHORA4
tmp(201) := STA & "100" & "100100100"; -- STA #4 @292
tmp(202) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(203) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(204) := JEQ & "000" & "011001010"; -- JEQ %SETDEZMIN
tmp(205) := STA & "111" & "111111110"; -- STA #7 @510
tmp(206) := LDA & "011" & "101000000"; -- LDA #3 @320
tmp(207) := GRT & "011" & "000000101"; -- GRT #3 @5
tmp(208) := JGT & "000" & "011001010"; -- JGT %SETDEZMIN
tmp(209) := STA & "011" & "100100011"; -- STA #3 @291
tmp(210) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(211) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(212) := JEQ & "000" & "011010010"; -- JEQ %SETUNIMIN
tmp(213) := STA & "111" & "111111110"; -- STA #7 @510
tmp(214) := LDA & "010" & "101000000"; -- LDA #2 @320
tmp(215) := GRT & "010" & "000001001"; -- GRT #2 @9
tmp(216) := JGT & "000" & "011010010"; -- JGT %SETUNIMIN
tmp(217) := STA & "010" & "100100010"; -- STA #2 @290
tmp(218) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(219) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(220) := JEQ & "000" & "011011010"; -- JEQ %SETDEZSEC
tmp(221) := STA & "111" & "111111110"; -- STA #7 @510
tmp(222) := LDA & "001" & "101000000"; -- LDA #1 @320
tmp(223) := GRT & "001" & "000000101"; -- GRT #1 @5
tmp(224) := JGT & "000" & "011011010"; -- JGT %SETDEZSEC
tmp(225) := STA & "001" & "100100001"; -- STA #1 @289
tmp(226) := LDA & "111" & "101100001"; -- LDA #7 @353
tmp(227) := CEQ & "111" & "000000000"; -- CEQ #7 @0
tmp(228) := JEQ & "000" & "011100010"; -- JEQ %SETUNISEC
tmp(229) := STA & "111" & "111111110"; -- STA #7 @510
tmp(230) := LDA & "000" & "101000000"; -- LDA #0 @320
tmp(231) := GRT & "000" & "000001001"; -- GRT #0 @9
tmp(232) := JGT & "000" & "011100010"; -- JGT %SETUNISEC
tmp(233) := STA & "000" & "100100000"; -- STA #0 @288
tmp(234) := LDI & "111" & "000000000"; -- LDI #7 $0
tmp(235) := STA & "111" & "100000000"; -- STA #7 @256
tmp(236) := RET & "000" & "000000000"; -- RET
```