Sistemas Reconfiguráveis – Eng. de Computação

Especificações para o quarto trabalho

1º semestre de 2025

1. Introdução

O microcontrolador PUC-251 é um controlador destinado a fins educacionais. Sua CPU é similar a um processador de 8 bits usada em pequenas aplicações embutidas, e seu conjunto de instruções é similar ao de um processador RISC.

O processador do PUC-251 usa memórias separadas para instruções e dados (arquitetura Harvard). A memória de instruções (programa) tem uma capacidade de 2k instruções, e a memória de dados é de 2048 x 8 bits. O processador também endereça 256 portas de entrada e saída de oito pinos. No processador existem quatro registradores de propósito geral de 8 bits, R0 a R3, sendo que o registrador R3 armazena os bits de status do processador: c (carry), z (zero), v (overflow). Também há um registrador de trabalho Wreg para operações com a ULA e movimentação de dados da memória e GPIO. Todas as instruções, exceto as instruções de desvio (JMP, CALL, SKIPC, SKIPZ, SKIPV e RET) são executadas em um único ciclo de *clock*. As instruções de desvio necessitam de dois ciclos de *clock*.

2. Objetivo

Implementar o microcontrolador PUC-251, empregando os módulos já desenvolvidos nos trabalhos anteriores e complementando com os blocos ainda não desenvolvidos, usando linguagem VHDL, ou instanciando, quando possível, macrofunções parametrizáveis, utilizando o recurso Mega Wizard Plugin Manager do *software* Quartus II. O topo da hierarquia deverá ser feito usando diagrama esquemático. Deverão ser instanciadas duas portas de entrada/saída. Uma delas (port_A) deverá ser configurada, por software, para funcionar com todos pinos configurados como saída. A outra (port_B) deverá ter todos os pinos configurados com entrada. Todas as entradas e saídas deverão ser do tipo std_logic ou std_logic_vector.

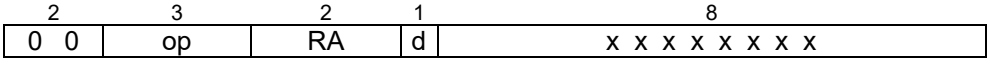
Deverá ser elaborado um programa para um teste simples de funcionamento. Esse programa deverá acionar sequencialmente os bits da porta de saída a intervalos regulares de aproximadamente 1 décimo de segundo. Essas saídas serão ligadas, no módulo DE2, a um conjunto de LEDs. Um bit da porta de entrada (que será ligado a uma chave no módulo) deverá alterar a sequência de acendimento dos LED's (do mais significativo para o menos significativo e vice-versa).

Para esse trabalho não será necessário o relatório. Deverá ser entregue um arquivo compactado (.zip ou .rar), com todos os arquivos dos projetos gerados no ambiente Quartus, incluindo o arquivo de simulação e o arquivo com o programa da aplicação. Assim como os outros trabalhos, a versão do Quartus a ser utilizada deve ser a 9.1sp2.

3. Código de instruções do processador

Todas as instruções do PUC-241 têm 16 bits de comprimento. O formato de codificação dessas instruções é variável, dependendo do tipo da instrução:

- **Instruções que envolvem a ALU e dois registradores (Reg-WReg)**



Esse formato é usado para instruções lógicas e aritméticas que operam valores armazenados em dois registradores. O primeiro operando é o registrador Wreg e o segundo operando pode ser quaisquer um dos quatro registradores R0 a R3. Os dois bits de RA indicam o registrador, de 00 para R0 até 11 para R3. O resultado da operação é armazenado em Wreg, se d=0, ou em RA, se d=1. Os valores de **op** definem a operação:

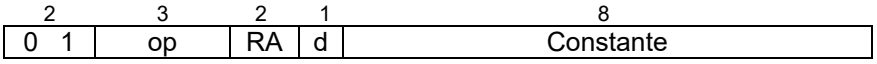
Instrução	Operação	op
ADD	Soma sem *carry in*	000
ADDC	Soma com *carry in*	001
SUB	Subtração sem *carry in* (Wreg – Rn)	010
SUBC	Subtração com *carry in* (Wreg – Rn)	011
AND	AND lógico bit a bit	100
OR	OR lógico bit a bit	101
XOR	XOR lógico bit a bit	110
MOV	Mover de um registrador para outro (Wreg → Rn)	111

O *flag* Z vai para '1' se o resultado da operação for zero, caso contrário vai para '0'.

Para ADD e ADDC, o *flag* C é o *carry out* (vai um) da adição. Para SUB e SUBC, o flag C é o *borrow out* da subtração e indica que a operação teve um resultado negativo. Para operações lógicas e para MOV, C é sempre ajustado para '0'.

O *flag* V só é atualizado nas operações aritméticas. Vai para '1' quando há um *overflow*, caso contrário vai para '0'. Um *overflow* ocorre quando: soma de dois números positivos resultada em um número negativo, a soma de dois números negativos temo como resultado um número positivo, um número positivo subtraído de um negativo resultando em um número negativo, ou um número negativo subtraído de um número positivo tendo como resultado um número positivo.

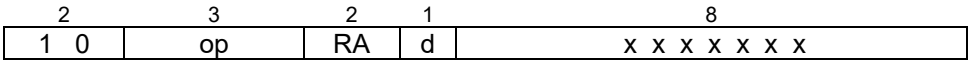
- ## Instruções que envolvem a ALU e um valor imediato (Reg-Immed)



Esse formato é usado para instruções lógicas e aritméticas que operam valores armazenados no registrador Wreg (d=0) ou um dos registradores RA (d=1) e um valor constante imediato, definido como parte do código da instrução. O resultado da operação é armazenado no registrador Wreg, se d=0, e em RA, se d=1. O comportamento dos *flags* é o mesmo para as instruções que envolvem a ALU e dois registradores.

Instrução	Operação	op
ADDI	Soma sem *carry in*	000
ADDCI	Soma com *carry in*	001
SUBI	Subtração sem *carry in* (Wreg ou Rn - valor)	010
SUBCI	Subtração com *carry in* (Wreg ou Rn - valor)	011
ANDI	AND lógico bit a bit	100
ORI	OR lógico bit a bit	101
XORI	XOR lógico bit a bit	110
MOVI	Mover valor imediato para o registrador	111

- **Instruções que envolvem a ALU e apenas um registrador**



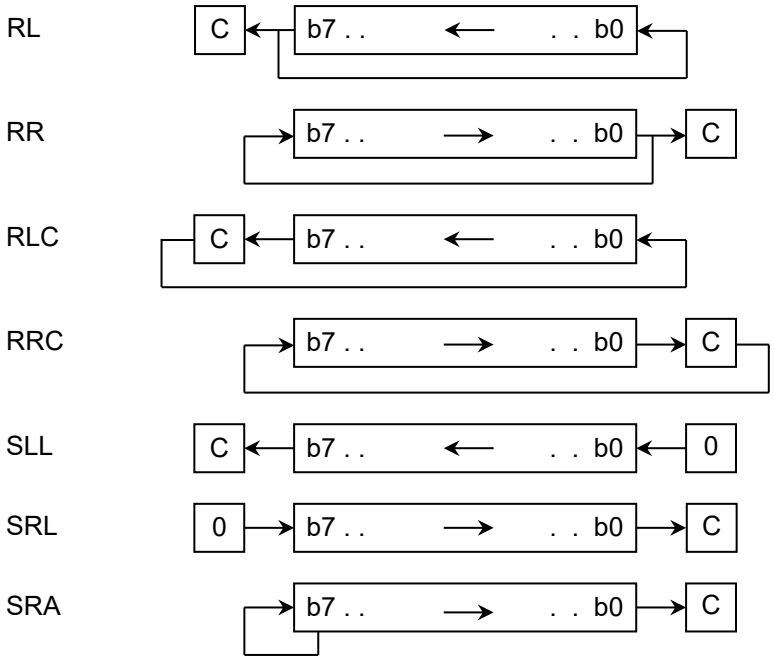
Esse formato é usado para instruções de complementar, deslocar e rodar que operam o valor do registrador Wreg (d=0), ou um dos registradores RA (d=1). O resultado da operação é armazenado no mesmo registrador escolhido. Os valores de **op** definem a operação:

Instrução	Operação	op
RL	Rotação para esquerda	000
RR	Rotação para direita	001
RLC	Rotação para esquerda através do *carry*	010
RRC	Rotação para direita através do *carry*	011
SLL	Deslocamento para esquerda lógico	100
SRL	Deslocamento para esquerda aritmético	101
SRA	Deslocamento para direita lógico	110
NOT	Complemento (inverte todos os bits)	111

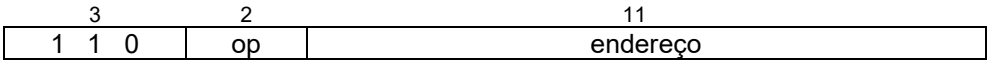
O *flag* Z vai para ‘1’ se o resultado da operação for zero, caso contrário vai para ‘0’.

O *flag* V não é modificado em nenhuma dessas instruções.

O *flag* C permanece inalterado na operação de complemento. Nas demais, é ajustado de acordo com o valor do bit de saída do deslocamento ou da rotação do operando, conforme o desenho abaixo:



- **Instruções de memória**

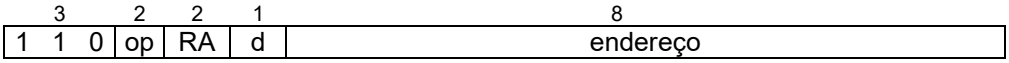


Esse formato é usado para instruções de carregar e armazenar na memória, envolvendo o registrador Wreg. O endereço é usado para acessar a memória. Para LDM, Wreg é o registrador de destino. Para STM, Wreg é o registrador de origem. Os valores de **op** definem a operação:

Instrução	Operação	op
LDM	Carrega um dado da memória para o registrador	00
STM	Armazena um dado do registrador na memória	01

Os *flags* Z, C e V não são afetados por essas instruções.

- **Instruções de E/S**

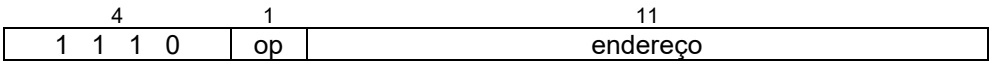


Esse formato é usado para instruções de carregar e armazenar nas portas de entrada e saída, envolvendo o um dos registradores RA ou o Wreg. O endereço é usado como o número da porta de E/S. Para INP, RA é o registrador de destino, se d=1, ou Wreg, se d=0. Para OUT, RA é o registrador de origem, se d=1, ou Wreg, se d=0. Os valores de **op** definem a operação:

Instrução	Operação	op
INP	Entrada da porta	10
OUT	Saída para porta	11

Os *flags* Z, C e V não são afetados por essas instruções.

- **Instruções de desvio incondicional e chamada de sub-rotina**

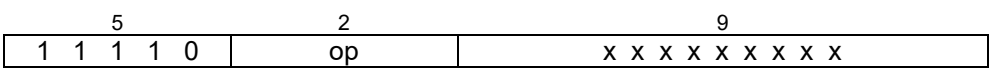


Esse formato é usado para instruções de desvio incondicional (JMP) ou chamada de sub-rotina (CALL). As duas instruções determinam a transferência da execução do programa para o endereço especificado na instrução, independentemente de qualquer condição. A diferença entre JMP e CALL é que essa última salva o endereço atual do PC na pilha (*stack push*). A sub-rotina deverá ser terminada com a instrução RET. Os valores de **op** definem a operação:

Instrução	Operação	op
JMP	Desvio incondicional	0
CALL	Chamada de sub-rotina	1

Os *flags* Z, C e V não são afetados por essas instruções.

- **Instruções de salto condicional**

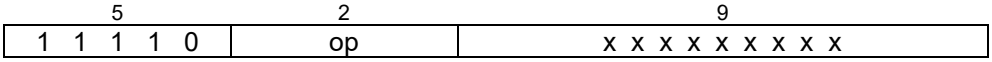


Esse formato é usado para instruções de salto da próxima instrução de forma condicional. Se a condição especificada pela instrução for verdadeira, o programa salta, sem executar, a próxima instrução na sequência do programa (executa novo *fetch*). Caso contrário, o programa continua normalmente com a instrução seguinte. Os valores de **op** definem a operação:

Instrução	Operação	op
SKIPC	Salta se *carry* (C = '1')	00
SKIPZ	Salta se zero (Z = '1')	01
SKIPV	Salta se *overflow* (V = '1')	10

Os *flags* Z, C e V não são afetados por essas instruções.

- ## Instrução de retorno

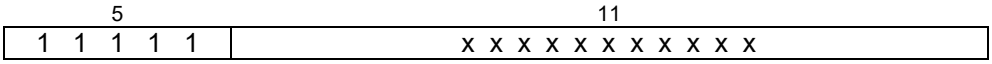


Esse formato é usado para instrução de retorno de sub-rotina (RET). Ao executar essa instrução, o endereço que foi salvo na pilha ao executar a instrução CALL deve ser restaurado ao contador de programa (*stack pop*) e deve ser feito novo ciclo de busca (executa novo *fetch*).

Instrução	Operação	op
RET	Retorno de sub-rotina	11

Os *flags* Z, C e V não são afetados por essas instruções.

- ## Instrução NOP



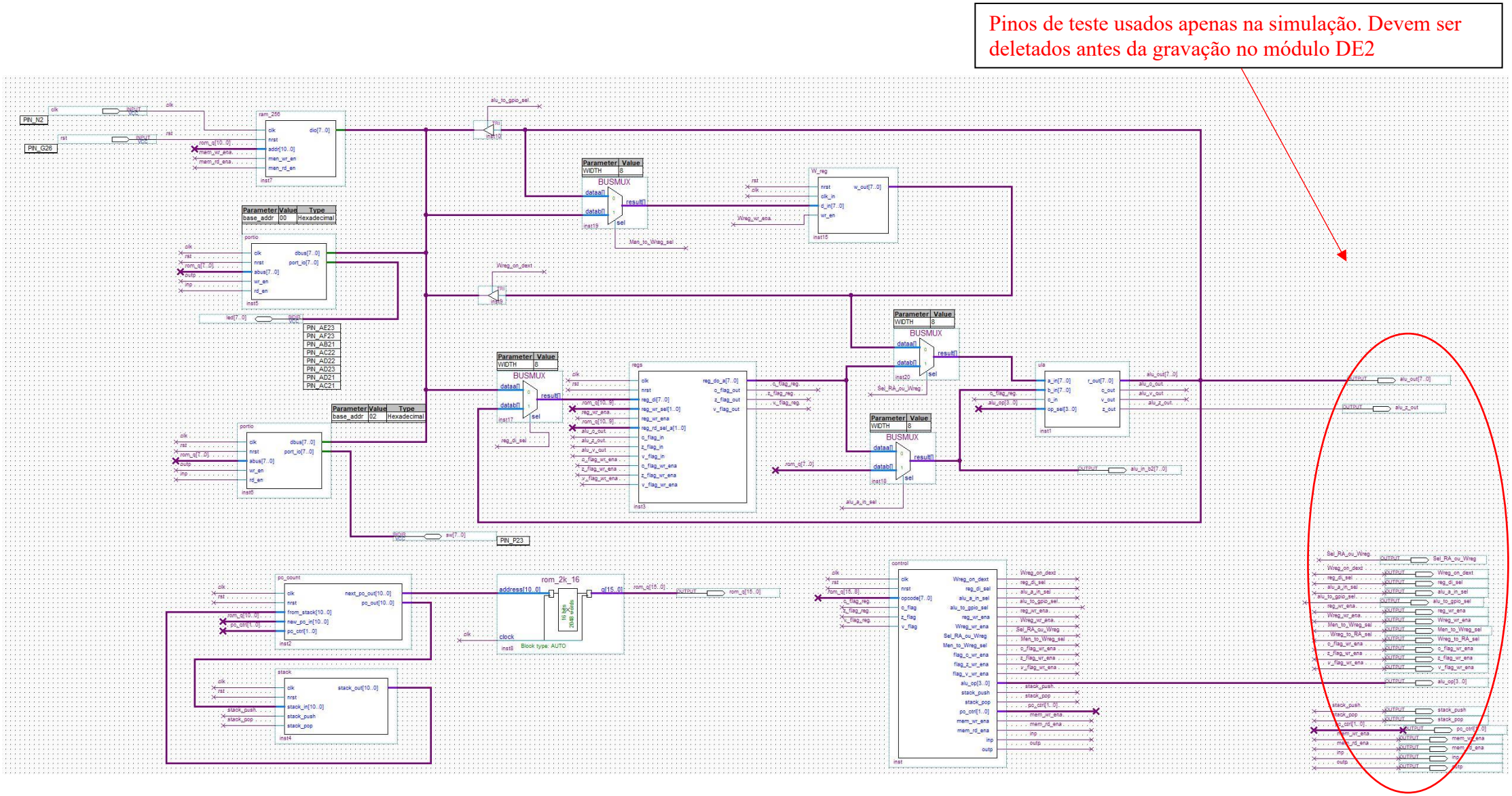
Essa instrução não tem efeito algum. O programa continua normalmente com a instrução seguinte. Nenhum *flag* é afetado por essa instrução.

Obs.: Todos os outros códigos de instrução não definidos devem corresponder à instrução NOP.

Dica 1: Observar que em todas as instruções que operam registradores, RA é sempre especificado pelos bits 10 e 9.

Dica 2: Observar que para a decodificação das instruções, o bloco de controle deverá receber os bits 15 a 8 da palavra de instrução.

Exemplo do topo da hierarquia:



Os blocos dos multiplexadores e da ROM deverão ser instanciados utilizando o recurso Mega Wizard Plugin Manager do Quartus II. O bloco de controle deverá ser desenvolvido usando linguagem VHDL. Os demais blocos já foram desenvolvidos nos trabalhos anteriores.