Instituição: Instituto Federal de Educação, Ciência e Tecnologia - Paraíba (IFPB).
Disciplina: Microprocessadores e microcontroladores**.**
Curso: Engenharia de Computação – 4º período
Professor: Fagner de Araujo Pereira.
Aluno (a): _____

Exercícios – Processador didático

1. Introdução

A Arquitetura do processador didático

Foi apresentado, durante as aulas, a arquitetura de um processador didático de 8 bits, capaz de executar instruções lógicas e aritméticas e operar sobre um conjunto de dados em um banco de registradores, possuindo uma porta de entrada e uma porta de saída.

O banco é formado por um conjunto de 12 registradores, sendo nomeados como R0, R1, R2, ..., R9, XL e XH. Os dados armazenados nos registradores podem ser externados a partir de dois barramentos de saída do banco, A e B, individualmente endereçados com 4 bits, [SBA_3:0] para o barramento A e [SBB_3:0] para o barramento B, no qual, por exemplo, o código binário 0b0000 acessa o primeiro registrador, R0, e segue até o código 0b1011, acessando o registrador XH.

É possível também fazer o armazenamento em um dos registradores de um dado resultante de uma operação lógica/aritmética, ou um dado proveniente da porta de entrada. A seleção da origem do dado é feita pelo bit [S_MUX_IN] do multiplexador MUX DATA_IN: Se o bit for 0, o dado é resultante de uma operação lógica ou aritmética, se o bit for 1, o dado vem da porta de entrada. O endereçamento do registrador de destino para armazenamento do dado é feito a partir dos bits [SRD_3:0]. Ainda é necessário a ativação do bit de habilitação de escrita no banco [LE]. Se esse bit estiver desabilitado, nenhum dado é armazenado no banco.

Um caso particular ocorre durante o armazenamento: Ao endereçar o registrador XL (0b1010), o registrador XH é simultaneamente endereçado. Isso ocorre porque os registradores XH e XL, quando usados no armazenamento de dados, funcionam como um único registrador de 16 bits destinado ao armazenamento dos resultados das multiplicações.

A ULA é responsável pela execução de 14 operações lógicas e/ou aritméticas com até dois operandos, A e B. Uma determinada operação é escolhida a partir dos 4 bits de seleção [SULA_3:0].

Na arquitetura do processador, há um multiplexador chamado MUX_CTE_IN, que permite que o operando A da ULA seja proveniente do barramento A do banco de registradores ou seja uma constante de 8 bits [CTE_7:0]. A seleção é feita diretamente pelo bit de seleção [S_MUX_CTE] do multiplexador: Se o bit for 0, o operando A vem do banco, se o bit for 1, o operando é a constante.

O resultado das operações da ULA pode ser de 8 ou 16 bits (no caso da multiplicação). O resultado pode ser diretamente armazenado no banco de registradores ou, ainda, ser direcionado para o registrador da porta de saída ROUT. Nesse caso, o bit de habilitação de escrita no registrador de saída [OE] deve ser setado.

O processador ainda fornece 8 bits que informam o status da operação atual, além de um registrador, chamado STATUS, que armazena tais bits. Os bits de status são setados para informar que: [CO] houve um estouro, [N] o resultado da operação da ULA é negativo, [Z] o resultado da operação da ULA é zero, [L] o operando A é menor que o B, [LE] o operando A é menor ou igual ao operando B, [E] o operando A é igual ao operando B, [GE] o operando A é maior ou igual ao operando B, e [G] o operando A é maior que o B.

A arquitetura foi concebida para que qualquer instrução seja executada em um único pulso de clock (*single clock*).

Os bits de controle do processador

Os 28 bits de controle do processador estão descritos a seguir:

[SBA_3:0] – Seleção do Registrador do Barramento A: selecionam o conteúdo de qual registrador será fornecido pelo barramento A do banco.

[SBB_3:0] – Seleção do Registrador do Barramento B: selecionam o conteúdo de qual registrador será fornecido pelo barramento B do banco.

[SRD_3:0] – Seleção do Registrador de Destino: seleciona o registrador no qual o dado de entrada do banco será armazenado quando o bit (LE) estiver habilitado.

[LE] – Load Enable: habilita, quando setado, o armazenamento do dado de entrada do banco no registrador endereçado pelos bits SRD_3:0.

[S_MUX_IN] – Seleção do multiplexador de entrada: seleciona se o dado de entrada do banco será o resultado da ULA ou o dado presente na porta de entrada do processador.

[SULA_3:0] – Seleção da ULA: seleciona a operação a ser realizada pela ULA.

[S_MUX_CTE] – Seleção do multiplexador do operando A da ULA: seleciona se o operando A da ULA será proveniente do barramento A do banco de registradores, quando resetado, ou se será a constante de 8 bits [CTE_7:0], quando setado.

[CTE_7:0] – Constante de 8 bits: é a constante de 8 bits que pode ser usada como operando A da ULA se o bit (S_MUX_CTE) estiver setado.

[OE] – Output Enable: habilita, quando setado, o armazenamento do dado de saída da ULA no registrador da porta de saída do processador.

A unidade de controle

A unidade de controle é responsável pelo gerenciamento dos 28 bits de controle para realizar a execução das instruções do processador. As instruções são armazenadas na memória de programa, que é endereçada pelo contador de programa (PC – Program Counter) a cada pulso de clock.

De forma simplificada, o barramento de dados da memória de programa poderia ser usado diretamente como os bits de controle do processador. Entretanto, em arquiteturas mais complexas, a quantidade de bits de controle é bem superior aos 28 aqui descritos, o que torna inviável usar essa estratégia. Assim, os bits de dados da memória de programa funcionam como a entrada de uma unidade decodificadora, cujas saídas são os bits de controle do processador. Essa unidade decodificadora é chamada de decodificador de instruções.

Além dos bits de controle descritos anteriormente, o decodificador de instruções também deve ser capaz de fornecer os bits de controle do contador de programa, para permitir que o fluxo de execução das instruções possa ser alterado, dependendo do tipo de instrução que estiver sendo executada, como nos casos das instruções de desvio condicional e incondicional.

O set de instruções do processador

A partir da definição da arquitetura, o processador apresentado foi dotado de um set de 32 instruções de uso genérico, cujos mnemônicos estão apresentados na lista abaixo:

***Instruções aritméticas e lógicas	Formato e quantidade de bits do código de máquina correspondente
ADD ra, rb, rc – [rc = ra + rb]	Opcode(5)+destino(4)+operandoA(4)+operandoB(4)
ADI ra, #cte – [ra = ra + #cte]	Opcode(5)+destino(4)+constante(8)
SUB ra, rb, rc – [rc = ra – rb]	Opcode(5)+destino(4)+operandoA(4)+operandoB(4)
SUBI ra, #cte – [ra = ra – cte]	Opcode(5)+destino(4)+constante(8)
AND ra, rb, rc – [rc = ra . rb]	Opcode(5)+destino(4)+operandoA(4)+operandoB(4)
ANDI ra, #cte – [ra = ra . #cte]	Opcode(5)+destino(4)+constante(8)
OR ra, rb, rc – [rc = ra | rb]	Opcode(5)+destino(4)+operandoA(4)+operandoB(4)
ORI ra, #cte – [ra = ra | #cte]	Opcode(5)+destino(4)+constante(8)
EOR ra, rb, rc – [rc = ra XOR rb]	Opcode(5)+destino(4)+operandoA(4)+operandoB(4)
EORI ra, #cte – [ra = ra XOR #cte]	Opcode(5)+destino(4)+constante(8)
NOT ra, rb – [rb = ~ra]	Opcode(5)+destino(4)+operando(4)
INC ra, rb – [rb = ra + 1]	Opcode(5)+destino(4)+operando(4)
DEC ra, rb – [rb = ra – 1]	Opcode(5)+destino(4)+operando(4)
MUL ra,rb – [rx = ra * rb]	Opcode(5)+operandoA(4)+operandoB(4)
MULI ra, #cte – [rx = ra * #cte]	Opcode(5)+operando(4)+constante(8)
SHL ra, rb – [rb = ra << 1]	Opcode(5)+destino(4)+operando(4)
SHR ra, rb – [rb = ra >> 1]	Opcode(5)+destino(4)+operando(4)

***Instruções de transferência de dados

MOV ra, rb – [rb = ra]	Opcode(5)+destino(4)+operando(4)
LDI ra, #cte – [ra = #cte]	Opcode(5)+destino(4)+constante(8)
IN ra – [ra = porta de entrada]	Opcode(5)+destino(4)
OUT ra – [porta de saída = ra]	Opcode(5)+operando(4)

***Instruções de desvio

JMP k – [PC = k]	Opcode(5)+constante(8)
RJMP k – [PC = PC + k]	Opcode(5)+constante(8)
BRL ra, rb, y_4 – [PC = PC+y se ra<rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRLE ra, rb, y_4 – [PC = PC+y se ra<=rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRE ra, rb, y_4 – [PC = PC+y se ra=rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRNE ra, rb, y_4 – [PC = PC+y se ra!=rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRGE ra, rb, y_4 – [PC = PC+y se ra>=rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRG ra, rb, y_4 – [PC = PC+y se ra>rb]	Opcode(5)+operandoA(4)+operandoB(4)+constante(4)
BRZ ra, k – [PC = PC+k se ra=0]	Opcode(5)+operando(4)+constante(8)

***Instruções de controle do processador

NOP – [nenhuma operação]	Opcode(5)
RESET – [PC=0]	Opcode(5)

Nas instruções acima, os seguintes termos são utilizados:
ra, rb e rc – representa um dos registradores do banco;
rx – é o registrador de 16 bits formado pela concatenação XH:XL;
#cte – é uma constante de 8 bits;
k – é uma constante de 8 bits;
y_4 – é uma constante de 4 bits.

Na memória de programa, cada instrução é representada por uma palavra binária de até 17 bits. A palavra na memória de programa pode ser representada da seguinte forma:

B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0

Os 5 bits mais significativos [B16:B12] formam o que é chamado de código da operação, ou *opcode*. Cada operação possui um *opcode* único. Por exemplo, a instrução **ADD** possui o *opcode* 0b00000, a instrução **ADI** possui o *opcode* 0b00001, e assim sucessivamente até a instrução **RESET** que tem o opcode 0b11111.

Os demais bits devem ser lidos sempre da esquerda para a direita e de acordo com a organização e quantidades apresentadas na lista de instruções apresentada. Os bits não utilizados em uma determinada instrução são ignorados pela unidade de controle, podendo assumir qualquer nível lógico.

2. Programando o processador didático

Considerando o set de instruções apresentado, escreva programas utilizando a linguagem Assembly do processador didático apresentado que execute as seguintes tarefas:

a) Leia o estado do bit 0 da porta de entrada e reproduza esse estado no bit 1 da porta de saída.

b) Faça com que 8 LEDs conectados à porta de saída acendam sequencialmente, um de cada vez, desde o bit 0 até o bit 7, em um loop infinito.

c) Faça com que os 4 bits menos significativos da porta de saída permaneçam setados, enquanto os 4 bits mais significativos permaneçam resetados enquanto o bit 3 da porta de entrada estiver setado. A saída deve funcionar de forma complementar enquanto o bit 3 da porta de entrada estiver desabilitado.

d) Considerando que os bits 0-3 da porta de entrada formem um dado X de 4 bits e os bits 4-7 formem um dado Y também de 4 bits, fazer com que a porta de saída exiba, simultaneamente, a operação lógica bit a bit **X+Y**, nos 4 bits menos significativos e **X.Y**, nos 4 bits mais significativos.

e) Exiba permanentemente na porta de saída uma contagem decrescente de 150 a 0.

f) Exiba na porta de saída o triplo do valor da porta de entrada.

g) Habilite o bit 7 da porta de saída se o valor da porta de entrada for ímpar e desabilite o mesmo bit, caso seja par.

h) Considerando que a porta de saída seja uma porta de comunicação paralela, envie através desta o texto “Hello World!”, sem as aspas.

i) Exiba na porta de saída a média aritmética dos 4 últimos valores lidos da porta de entrada.

j) Transcreva o seguinte código em linguagem C:

```
void main( )
{
    int A=0;
    int B=0;

    while(TRUE)
    {
        A = input_;
        B = 5*A;
        if(A > 10)
            soma( );
        else
            subtrai( );
     }
}

soma( )
{
   output_ = B+A;
{

subtrai( )
{
   output_ = B-A;
{
```

obs.: os comandos input_ e output_ correspondem à operações de leitura e escrita nas portas de entrada e saída, respectivamente.

Dúvidas: fagnereng@gmail.com