Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2016/1
Prof. Marcus Vinicius Lamar
Equipes de até 5 pessoas.
Entrega do relatório (pdf) e fontes em um arquivo zip pelo Moodle até dia 02/05/2016

Laboratório 1 – PARTE A
- Assembly MIPS –

Objetivos:

- Familiarizar o aluno com o Simulador/Montador MARS;
- Desenvolver a capacidade de codificação de algoritmos em linguagem Assembly MIPS;
- Desenvolver a capacidade de análise de desempenho de algoritmos em Assembly;

(2.0) 1) Simulador/Montador MARS

Instale em sua máquina o simulador/montador MARS v.4.5 Custom 4 disponível no Moodle.

(0.0)1.1) Dado o programa sort.s e o vetor: V[10]={5,8,3,4,7,6,8,0,1,9}, ordená-lo em ordem crescente e contar o número de instruções (por tipo e por estatística) exigido pelo algoritmo. Qual o tamanho em bytes do código executável?

(0.0)1.2) Explique o que é o valor 589834 armazenado a partir do endereço 0x10010028. O emulador trabalha no formato Little Endian ou Big Endian?

(2.0)1.3) Considerando a execução em um processador MIPS com frequência de *clock* de 50MHz, que necessita 1 ciclo de *clock* para a execução de cada instrução (CPI=1).

(1.0) a) Calcule o tempo de execução necessário para o programa e dados do item 1.1).

(1.0) b) Para os vetores de entrada de n elementos já ordenados {1,2,3,4...n} e ordenados inversamente {n, n-1, n-2,...,2,1}, meça a o número de instruções necessário a execução para n={1,2,3,4,5,6,7,8,9,10,20,30,40,50,60,70,80,90,100} e plote esses dados em um mesmo gráfico n x I ;

(3.0) 2) Compilador GCC

Instale na sua máquina o *cross compiler* MIPS GCC disponível no Moodle.
Forma de utilização: mips-sde-elf-gcc –S teste.c #diretiva –S para gerar o arquivo Assembly teste.s
Inicialmente, teste com programas triviais em C para entender a convenção utilizada para a geração do código Assembly.

(1.0)2.1) Dado o programa sort.c, compile-o e comente o código em Assembly obtido indicando a função de cada uma das diretivas do montador usadas no código Assembly (.file .section .mdebug .previous .nan .gnu_attribute .globl .data .align .type .size .word .rdata .align .ascii .text .ent .frame .mask .fmask .set).

(0.5) 2.2) Indique as modificações necessárias no código Assembly gerado para poder ser executado corretamente no Mars.

(1.0)2.3) Compile novamente o programa sort.c e compare o número de instruções executadas e o tamanho em bytes dos códigos obtidos com os dados do item 1.1) para cada diretiva de otimização da compilação {-O0, -O1, -O2, -O3, -Os}.

(5.0) 3) Primos Gêmeos

Dois números naturais primos são ditos gêmeos se diferença entre eles for 2.
Ex.: PG_1=(3,5) PG_2=(5,7), PG_3=(11,13), PG_4=(17,19), ..., PG_i=(P_i,P_i+2), ..

(1.0)3.1) Escreva um procedimento em Assembly MIPS que retorne o primeiro elemento do i-ésimo par de números primos gêmeos, int PG(int i)

(1.0)3.2) Escreva um programa void main(void) que leia o valor de i do teclado e mostre na tela o resultado com a formatação

printf(“Primos Gêmeos(%d)=(%d,%d)\n”,i,P[i],P[i]+2)

(1.0)3.3) Encontre o número i_{max}, de forma que P_{imax}+2 seja o maior número Primo Gêmeo representável utilizando uma palavra de 32 bits sem sinal.

(1.0)3.4) Para o seu procedimento, determine uma equação NInstr(i) que seja capaz de estimar o número de instruções necessárias ao cálculo de P_i+2. Desenhe as curvas de NInstr(i) e do número de instruções real em um gráfico para i={1,2,3,....,100}.

(1.0)3.5) Considerando que vamos utilizar a implementação de um processador MIPS32 com CPI de 1 e frequência de *clock* de 50MHz, qual o tempo esperado para o cálculo do valor de P_{imax}+2? Qual deveria ser a frequência deste processador para que o cálculo do P_{imax}+2 se realizasse em 1 minuto?

Universidade de Brasília
Departamento de Ciência da Computação

Disciplina: CIC 116394 – Organização e Arquitetura de Computadores – Turma A 2016/1
Prof. Marcus Vinicius Lamar

Laboratório 1 – PARTE B
– Síntese de hardware –

Objetivos:

- Introduzir ao aluno a Linguagem de Descrição de Hardware Verilog;
- Familiarizar o aluno com a plataforma de desenvolvimento FPGA DE2 da Altera e o software QUARTUS-II;
- Desenvolver a capacidade de análise e síntese de sistemas digitais usando HDL;

(0.0) 1) Implementação de um driver para display de 7 segmentos

(0,0)1.1) Crie um novo diretório e novo projeto Verilog. Defina o novo projeto conforme descrito no arquivo Set.txt Inclua o arquivo do decodificador assíncrono (decoder7.v); Compile;
(0,0)1.2) Realize a verificação funcional e temporal do projeto decodificador por simulação com forma de onda; Qual o maior tempo de propagação (tpd) estimado para o projeto?
(0,0)1.3) Crie uma versão síncrona do decodificador (decoder7.v); Compile;
(0,0)1.4) Realize a verificação funcional e temporal do projeto decodificador por simulação com forma de onda; Quais os maiores tempo de setup (tsu), tempo de hold (th) e atraso da saída para o clock (tco) estimado para o projeto? A saída está correta?
(0,0)1.5) Inclua o arquivo de teste (teste.v) com as interconexões com a placa DE2-70, sintetize e teste;

(10,0) 2) Implementação de Somador de 4 bits

(2,0)2.1) Implementação por desenho esquemático no Quartus-II (conforme visto em CD - 116351):
(0.5)a) Crie um novo diretório e projeto esquemático (somador_esquematico). Implemente os blocos esquemáticos do meio-somador de 1 bit (half_add.bsf), do somador completo de 1 bit (full_add.bsf) e do somador completo de 4 bits (add4.bsf), conforme as figuras em anexo;
(1.0)b) Realize a simulação funcional completa do projeto do somador de 4 bits; Realize a simulação temporal com T=10ns para cada alteração de A. Explique o que ocorre;
(0.5)c) Crie o arquivo de teste (teste.bdf) com as interconexões com a placa DE2-70, sintetize e teste;

(2,0)2.2) Descrição Verilog ao nível de portas lógicas no Quartus-II:
(0.5)a) Crie um novo diretório e projeto Verilog (somador_portas). Implemente as descrições do meio-somador de 1 bit (half_addv.v), do somador completo de 1 bit (full_addv.v) e do somador completo de 8 bits (add8v.v), conforme as figuras;
(1.0)b) Realize a simulação funcional completa do projeto do somador de 8 bits; Realize a simulação temporal com T=10ns para cada alteração de A. Explique o que ocorre.
(0.5)c) Crie o arquivo de teste (teste.v) com as interconexões com a placa DE2-70, sintetize e teste;

(2,0)2.3) Descrição Verilog ao nível comportamental no Quartus-II:
(0.5)a) Crie um novo diretório e projeto Verilog (somador_comportamental). Implemente um somador completo de 4 bits (add4cv.v)
Dica: assign {Carry,Sum}=A+B;
(1.0)b) Realize a simulação funcional completa do projeto do somador de 4 bits; Realize a simulação temporal com T=10ns para cada alteração de A. Explique o que ocorre.
(0.5)c) Modifique o arquivo de teste (teste.v) com as interconexões com a placa DE2-70, sintetize e teste; Apresente a verificação do projeto. Filme e narre os testes. Poste o vídeo no YouTube com o link no relatório.

(2,0)2.4) Análise comparativa das três sínteses do somador de 4 bits:
(1.0)a) Compare os números de Elementos Lógicos necessários e os maiores tempos de atraso de propagação (tpd) para cada implementação.
(1.0)b) Implemente as correspondentes versões síncronas. Quais as maiores frequências de clock utilizáveis?

(2,0) 3) Análise comparativa das sínteses do somador de 32 bits:
Repita os procedimentos 2.1) 2.2) e 2.3) de forma a criar 3 versões assíncronas de um somador de 32 bits:
(1.0)a) Compare os números de Elementos Lógicos necessários e os maiores tempos de atraso de propagação (tpd) para cada implementação.
(1.0)b) Implemente as correspondentes versões síncronas. Quais as maiores frequências de clock utilizáveis?

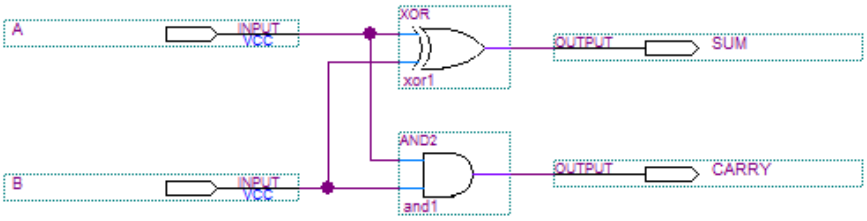

Figura 1- half_add.bsf

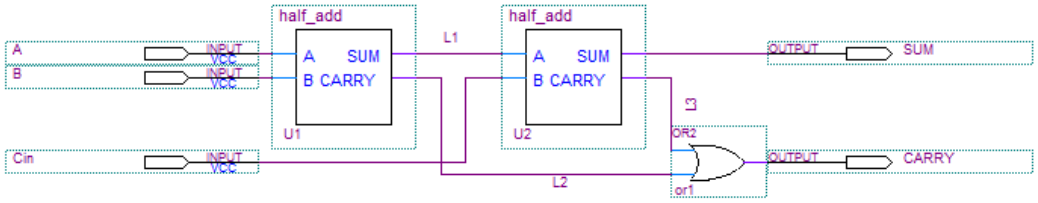

Figura 2- full_add.bsf

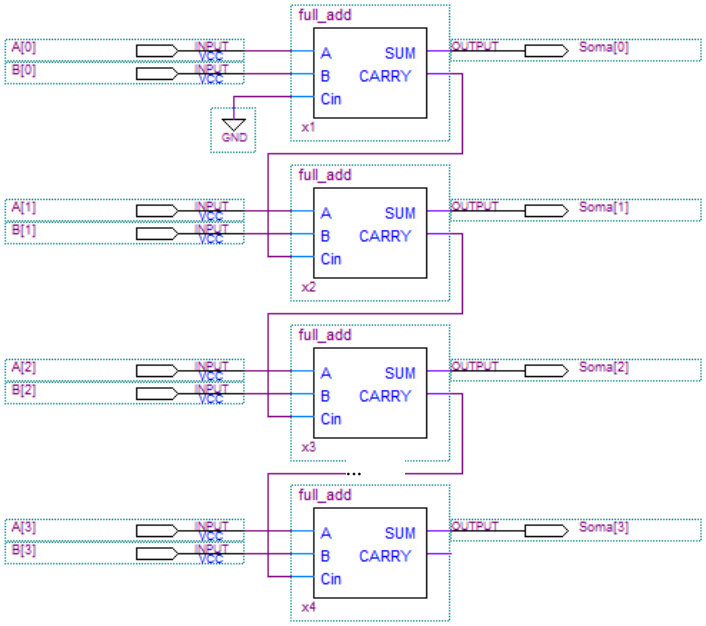

Figura 3- add4.bsf

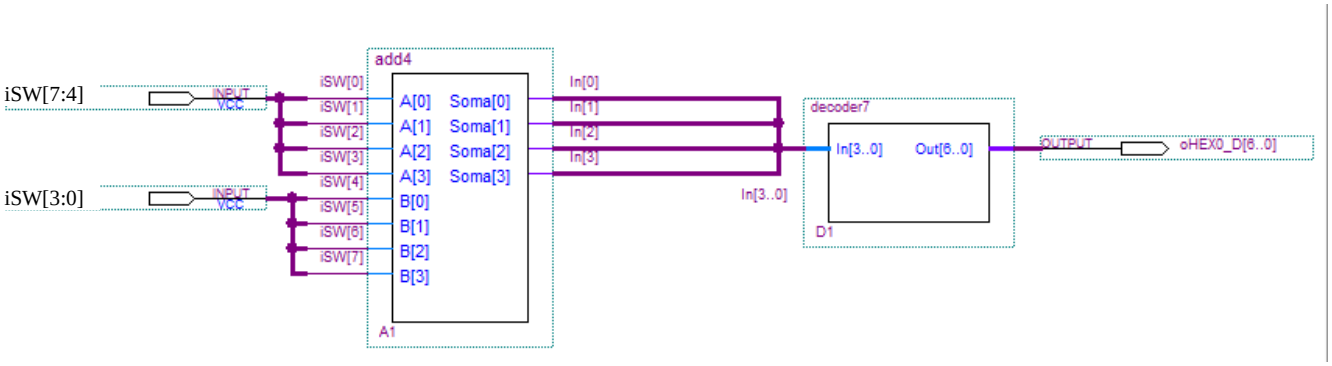

Figura 4 - teste.bdf