UNIVERSIDADE ESTADUAL DE FEIRA DE SANTANA

DEPARTAMENTO DE TECNOLOGIA

TEC499 SISTEMAS DIGITAIS

RELATÓRIO TÉCNICO DO PRIMEIRO PROBLEMA

Ivan dos Santos Peixoto, Washington Pagotto Batista, Pedro Kennedy Silva

Tutor: Anfranserai Morais Dias

1 INTRODUÇÃO

Este documento descreve algumas importantes características do processador Nios II, do emulador JNiosEmu e também especifica os programas de testes dos algorítmos Bubble Sort, Potência, Gerador de Números Primos, Sequência de Fibonacci e Cálculo Fatorial. Ainda apresenta, nos capítulos3 e 4, dados e instruções dos testes do programa.

2 PRINCIPAIS CARACTERÍSTICAS DO PROCESSADOR NIOS II

As instruções no Nios II são formadas por 32 bits, sendo executadas diretamente pelo processador, além de haver também um conjunto de instruções que incluem uma quantidade de pseudo instruções que podem ser usadas em linguagem assembly. O montador substitui cada pseudo-construção por uma ou mais instruções da máquina.

2.1 REGISTRADORES DE PROPÓSITO GERAL

O processador possui um conjunto de 32 registradores de propósito geral. As principais características e seus respectivos significados serão apresentados na Tabela 1 a seguir.

Registrador	Nome	Registrador	Nome	Registrador	Nome
0	r0	11	r11	22	r22
1	at	12	r12	23	r23
2	r2	13	r13	24	et
3	r3	14	r14	25	bt
4	r4	15	r15	26	gp
5	r5	16	r16	27	sp
6	r6	17	r17	28	fp
7	r7	18	r18	29	ea
r8	r8	19	r19	30	ba
r9	r9	20	r20	31	ra
r10	r10	21	r21	-	-

Tabela 1 – Relação dos registradores do Nios II.

2.2 ARQUITETURA RISC

A arquitetura RISC é constituída por um pequeno conjunto de instruções simples que são executadas diretamente pelo hardware, sem a intervenção de um interpretador (microcódigo), ou seja, as instruções são executadas em apenas uma microinstrução.

A pouca variedade dos tipos de instrução e dos modos de endereçamento, além de demandar uma Unidade de Controle mais simples, também traz outro importante benefício, que é a previsibilidade. Como as intrusões variam pouco de uma para outra, é mais fácil para a Unidade de Controle prever quantos ciclos serão necessários para executá-las. Esta previsibilidade traz benefícios diretos para o ganho de desempenho com o Pipeline.

2.3 MODELO ISA

O campo OP, na instrução do Nios II, especifica a classe principal de um opcode. A maioria dos valores de OP são codificações para instruções de tipo I. Uma codificação, OP = 0x00, é a chamada de instrução Jtype. Outra codificação, OP = 0x3a, é usada para todas as instruções do tipo R.Todas as codificações indefinidas de OP e OPX são reservadas.

2.3.1 Formato do tipo R

Em operações do tipo R todos os argumentos são tidos como registros. As instruções do tipo R são formadas por 6 bits OPcode OP, três campos de registro (A, B e C) de 5 bits cada e um campo de OPcode OPX de 11 bits. Os campos A e B de registro, normalmente são contem operandos de origem, sendo o campo C o de destino. As instruções do tipo R incluem operações aritméticas e lógicas como add e nor; Operações de comparação.

Opcode	A	B	C	Opx
6 bits	5 bits	5 bits	5 bits	1q bits

Tabela 2 – Formato do tipo R

2.3.2 Formato do tipo I

Operações do tipo I são caracterizadas por possuírem um valor imediato incluído na instrução. Instruções do tipo I possuem campo OPcode (6 bits), dois campos de registro (5 bits A e B) e um campo de dados imediatos (16 bits IMM16).

OPCode	A	B	IMM16
6 bits	5 bits	5 bits	16 bits

Tabela 3 – Formato do tipo I

2.3.3 Formato do tipo J

As instruções do tipo J contêm um campo de opcode de 6 bits e um campo de dados imediatos de 26 bits. Instruções do tipo J, como chamada e jmpi, transferem execução em qualquer lugar dentro de uma faixa de 256 MB.

Opcode	IMM26
6 bits	26 bits

Tabela 4 – Formato do tipo J

2.4 JNIOSEMU

O JNiosEmu é um emulador e ambiente de desenvolvimento com o propósito de uso para programação em assembly.O JNiosEmu emula um sistema de computador simples que inclui registros, memória e dispositivos de entrada e saída básicos que permite que a execução de instruções do assembler. Ele oferece considerável comodidade na montagem e execução do código e ainda exibe os resultados das ações do mesmo.

2.5 PORTA SERIAL

Através do emulador Jniosemu são inseridos os dados na porta serial por meio das entradas uart0 e uart1. Mas para a realização desse projeto só foi utilizada a uart0. Como pode ser visto na Tabela 6, apresenta portas que fazem a leitura, escrita e a verificação feita pelo próprio processador se existe alguma leitura ou escrita na porta serial.

Para facilitar a manipulação da escrita e leitura na porta serial, o emulador fornece bibliotecas capazes de auxiliá lo, como apresentado na tabela 1. Vale ressaltar, que a entrada serial se baseia em tabela ascii e só aceita um caracter por vez.

0x860	0x864	0x868
Leitura	Escrita	Verifica se existe alguma leitura ou escrita

Tabela 5 – Portas da entrada serial

Rotina	Descrição
Nr_uart_rxchar	Lê um caracter do uart
Nr_uart_txchar	Imprime um único caracter
Nr_uart_txhex	Imprime um valor inteiro em hexadecimal
Nr_uart_txhex16	Imprime um valor inteiro curto em hexadecimal
Nr_uart_txhex32	imprime um valor inteiro longo em hexadecimal
Nr_uart_txstring	Imprime uma String

Tabela 6 – Portas da entrada serial

Como a porta serial é em ascii e a entrada é feita de um caracter por vez. É necessário manipular o assembly para que aceite números maiores que 9 e faça sua devida operação em decimal. Ressaltando que ao entrar com número 1 na uart0, o valor armazenado no registrador será 49, que é seu correspondente na tabela ascii, sendo necessário uma subtração para chegar ao valor desejado.

3 FUNCIONAMENTO DOS TESTES

Para cada programa teste é definido um tipo de entrada diferente na porta serial, especificados na tabela 7.

Teste	Fibonacci	Bubble Sort	Fatorial	Potência	Primos
Entrada	2.	3.5.6.7.8#	5.	2.5#	4.

Tabela 7 – Formatos das entradas na porta Serial

A finalização do número inteiro dos testes de Fibonnacci, Fatorial e Números Primos é o . (ponto), definido na tabela ascii pelo número 46.

Já no BubbleSort, que é a inserção de um vetor de tamanho fixo cinco, o separador dos valores é o . (ponto) e a finalização desse vetor é o # (cerquilha).

Na potência o . (ponto) separa a base, que é o primeiro elemento, do expoente e para sinalizar sua finalização usa-se # (cerquilha).

4 ESPECIFICAÇÕES DOS TESTES

Teste	Fibonacci	Bubble Sort	Fatorial	Potência	Primos
Entrada	um valor inteiro	5 valores inteiros para o vetor	um valor inteiro	dois valores inteiros	um valor inteiro
Saídas	hexadecimal	Vetor ordenado em hexadecimal	Hexadecimal	Hexadecimal	Hexadecimal até 32 bits
Limitações	Para cada valor, 3 espaços no stack são ocupados, e no ultimo nenhum, o limite da memória é 32	vetor fixo de 5 elementos de até 32 bits	só faz o cálculo até fatorial de 10 até 32 bits	o resultado não ultrapassa um hexadecimal de 32 bits	só gera números primos até 32

Tabela 8 – Especificações dos Testes

5 CASOS TESTADOS

A seguir na Tabela 9 são exibidos casos de testes aplicados, e resultados encontrados (intervalos de valores aplicados).

6 RESULTADOS ALCANÇADOS

Os resultados apresentados foram satisfatórios, com exceção apenas do armazenado total da sequência de Fibonnaci que só foi exibido o seu resultado final. Poderia ser feito o armazenamento em words de tamanho 64 ou em uma pilha.

A escolha de definir um tamanho fixo do vetor no teste BubbleSort, foi da equipe visto a dificuldade de manipulação de words. Sendo assim, uma melhoria para esse teste seria informar na entrada, além dos valores do vetor, o tamanho de seu vetor. Assim utilizando o comando skip 256 na word, criaria 64 espaços vazios para serem utilizados. Com essa melhoria seria possível fazer um vetor não fixo e de até 64 valores.

Outra melhoria na forma de armazenação do teste de número primo, seria a utilização de words visto que é possível armazenar o dobro da pilha. O mesmo caso se aplica ao teste fatorial sendo que com a utilização de words seria possível o fatorial de até 21, caso não ultrapassasse o tamanho em hexadecimal de 32 bits. Uma outra melhoria do Fatorial seria mostrar todo o caminho percorrido até chegar o resultado .

Teste	Intervalo	Motivo	Nome do teste	Resultado
Primos	1-32	Stack da memória possui apenas 32 espaços	32	2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131
Fatorial	1-12	Fatorial de 13 em diante ocupa mais de 32 bits	12	479001600
Fibbonaci	1-11	Para cada valor, 3 espaços no stack são ocupados, e no ultimo nenhum, o limite da memória é 32	11	89
Potência	1Î- 655352̂	Limite de 32 bits, valor aproximado máximo é este, limite correto seria = 65656,982̂	655352̂	4294836225
Bubblesort	5 valores máximos	-	33, 55, 7, 43, 49 *	7, 33, 43, 49, 55

Tabela 9 – Casos testados