

Laboratório 1

Usando o processador Nios II e recursos do Altera Monitor

Este primeiro laboratório envolve o processador Nios II e sua linguagem de montagem. Vamos usar um sistema básico chamado de *DE2 Basic Computer*, o qual inclui o processador Nios II, memória e alguns periféricos de entrada e saída. O programa *Altera Monitor* é usado para mostrar os processos de compilação, carregamento e execução das aplicações na placa DE2.

Parte I

A partir de agora vamos explorar algumas características do programa Altera Monitor através de uma aplicação escrita em linguagem de montagem do Nios II. Considere o programa na Figura 1, o qual encontra o maior número em uma lista de inteiros de 32 bits armazenada na memória. Para o laboratório, você vai precisar reescrever o programa da Figura 1 usando algum editor de texto.

```
/* Program that finds the largest number in a list of integers */
.equ LIST, 0x500          /* Starting address of the list */

.global _start
_start:
    movia r4, LIST        /* r4 points to the start of the list */
    ldw r5, 4(r4)         /* r5 is a counter, initialize it with n */
    addi r6, r4, 8        /* r6 points to the first number */
    ldw r7, (r6)          /* r7 holds the largest number found so far */
LOOP:
    subi r5, r5, 1        /* Decrement the counter */
    beq r5, r0, DONE     /* Finished if r5 is equal to 0 */
    addi r6, r6, 4        /* Increment the list pointer */
    ldw r8, (r6)          /* Get the next number */
    bge r7, r8, LOOP     /* Check if larger number found */
    add r7, r8, r0        /* Update the largest number found */
    br LOOP
DONE:
    stw r7, (r4)         /* Store the largest number into RESULT */
STOP:
    br STOP              /* Remain here if done */

.org 0x500
RESULT:
    .skip 4              /* Space for the largest number found */
N:
    .word 7               /* Number of entries in the list */
NUMBERS:
    .word 4, 5, 3, 6, 1, 8, 2 /* Numbers in the list */
.end
```

Figura 1: Programa em linguagem de montagem para encontrar o maior número.

Note que uma lista com números é fornecida como exemplo. Esta lista começa no endereço hexadecimal 500, como especificado pela diretiva **.org** do montador. A primeira palavra (4 bytes) é reservada para armazenamento do resultado, que será o maior número encontrado. A próxima palavra especifica o número de entradas na lista. As palavras seguintes contêm os números da lista.

Certifique-se que você entenda o programa da Figura 1 e o significado de cada instrução. Note o uso extensivo de comentários no programa. Procure sempre usar comentários úteis em seus programas!

Prossiga com os seguintes passos:

1. Crie um novo diretório; escolhamos como exemplo o nome *lab1_part1*. Copie o arquivo que você editou com o programa da Figura 1 para dentro deste diretório.
2. Use o programa Altera Monitor para criar um novo projeto neste diretório; escolhamos como nome *part1*. Escolha a opção **Assembly Program** quando for perguntado, mas não escolha nenhum programa exemplo. Clique em **Next**.
3. Você deve especificar o nome do arquivo que contém o programa em linguagem de montagem. Clique em **Add** e na janela que aparecer especifique o nome do arquivo que você editou, e sua localização. Clique em **Next** até chegar na janela de seleção do dispositivo de memória. Certifique-se que o dispositivo de memória selecionado seja a SDRAM. Note que o campo *Start offset in device* será 0, porque o programa da Figura 1 não indica que ele deva ser carregado em uma posição diferente da posição padrão 0. Clique em **Finish**.
4. Compile e carregue o programa.
5. O programa monitor vai mostrar uma visão do código carregado na memória desmontado, como indicado pela Figura 2. Note que a pseudo-instrução **movia** do programa original foi substituída por duas instruções nativas, **orhi** e **addi**, as quais carregam o endereço de 32 bits de LIST para o registrador *r4* em duas partes de 16 bits (porque um operando imediato está restrito a 16 bits). Examine o código desmontado e veja a diferença em relação ao código fonte original. Certifique-se que você entenda o significado de cada instrução. Também observe que seu programa foi carregado na posição de memória iniciando-se no endereço 0. Este endereço corresponde à memória SDRAM, que foi selecionada quando da especificação dos parâmetros do sistema.
6. Execute o programa. Quando o programa estiver rodando, você não verá modificação alguma (tal como conteúdo dos registradores e da memória) na janela do monitor, porque o programa monitor não pode se comunicar com o processador na placa DE2. Mas, se você parar o programa o estado atual destes componentes será mostrado. Faça isso e observe que o programa parou de executar na última instrução de salto carregada no endereço de memória 0x34. Note que o maior inteiro encontrado na lista de exemplo é 8, como indicado pelo conteúdo do registrador *r7*. Este valor é também armazenado na posição de memória 0x500, que pode ser vista ao abrir a tábua de memória na janela do programa monitor.

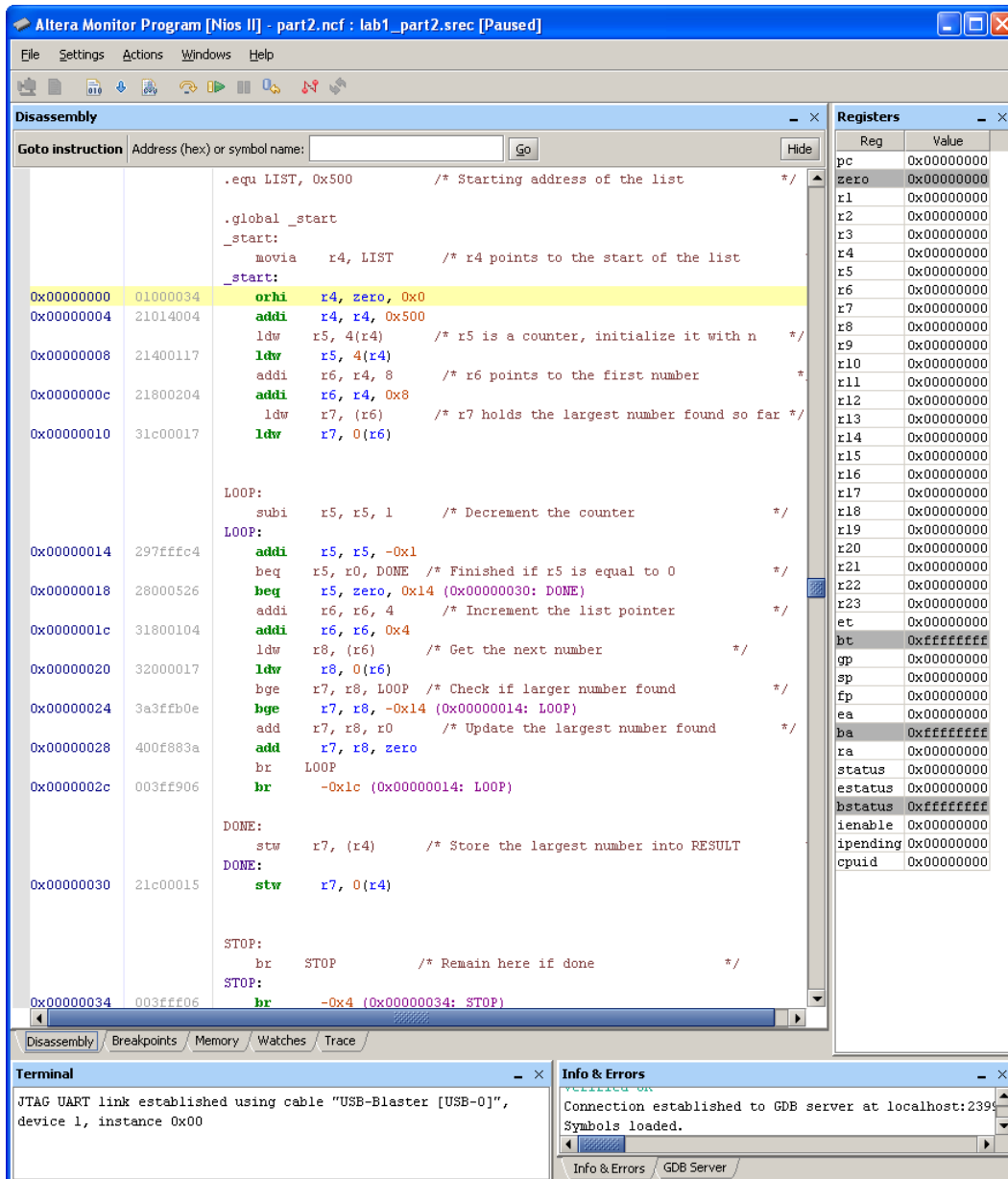



Figura 2: A visão do programa da Figura 1 com código desmontado.

7. Retorne ao início do programa clicando no ícone . Agora use a opção *single step* para executar instrução por instrução clicando no ícone . Observe como cada instrução altera o conteúdo dos registradores do processador.
8. Coloque o valor do contador de programa (*Program Counter*) para 0. Note que esta ação tem o mesmo efeito que clicar no ícone de *restart*. .
9. Desta vez adicione um *breakpoint* no endereço 0x28 (clicando na barra cinza à esquerda deste endereço) de forma que o programa vai automaticamente parar sua execução quando a instrução de salto nesta posição estiver prestes a ser executada. Execute o programa e observe o conteúdo do registrador *r7* cada vez que o breakpoint é atingido.

10. Remova o breakpoint (clcando sobre ele). Agora, faa o contador de programa ficar com o valor 0x8, o que desconsidera as primeiras duas instrues que carregam o endereo de LIST para o registrador *r4*. Faa, tambem, com que o valor do registrador *r4* se torne 0x504. Execute o programa clicando no icone . Qual ser o resultado desta execuo?

Parte II

Nesta parte voc deve escrever um programa em linguagem de montagem do Nios II para gerar os primeiros *n* nmeros da srie de Fibonacci (iterativamente). Nesta srie, os primeiros dois nmeros so 0 e 1, e cada nmero subsequente  gerado adicionando-se os dois ltimos nmeros da srie. Por exemplo, para *n* = 8, a srie 

0, 1, 1, 2, 3, 5, 8, 13

Seu programa deve armazenar os nmeros em posies sucessivas de memria, comeando no endereo 0x1000. Coloque o valor de teste *n* na posio 0xffc.

Prossiga como a seguir:

1. Crie um diretrio, *lab1_part2*.
2. Escreva um programa em linguagem de montagem que compute a srie de Fibonacci requisitada, e coloque o arquivo fonte no diretrio *lab1_part2*.
3. Use o programa monitor para criar um novo projeto, *part2*, e especifique que seu programa deve usar o *DE2 Basic Computer*.
4. Execute seu programa.
5. Examine as posies de memria a partir de 0x1000 para verificar que seu programa est correto.

O que incluir no seu relatrio

Faa um resumo das atividades das partes I e II, vistas neste laboratrio. No seu relatrio:

- explique o que aconteceu na execuo do item 10 da parte I;
- o uso da diretiva `.org` no cdigo da Figura 1  necessrio? Qual sua funo?
- existe um erro sutil no cdigo da Figura 1. Qual  esse erro? O que voc faria para consert-lo?
- coloque o algoritmo para a parte II.

No esquea de enviar junto com seu relatrio todo o cdigo fonte desenvolvido neste laboratrio. Voc pode compactar os arquivos em um nico e anex-lo no portflio do seu grupo no TelEduc. No se esquea de compartilh-lo somente com o formador!