Laboratório 1

Usando o processador Nios II e o CPUlator Nios II System Emulator

Este primeiro laboratório envolve o processador Nios II e sua linguagem de montagem. Para a atividade será usado o simulador CPUlator configurado para um sistema com o Nios II. Você deve entregar os arquivos fontes que você escrever neste laboratório, assim como um arquivo texto respondendo as perguntas feitas durante as atividades (numeradas e em vermelho).

Parte I

A partir de agora vamos explorar algumas características do CPUlator Nios II 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. Para isso você pode utilizar diretamente o editor do CPUlator ou então utilizar qualquer outro editor de sua preferência e carregar o arquivo fonte através da interface do CPUlator.

```
/* 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. Escreva o programa conforme especificado pela Figura 1. Procure estudar e entender as diretivas/instruções conforme você escreve o código. Não esqueça de salvar o arquivo frequentemente para não perder o conteúdo.

2. Compile e carregue o programa (F5) (botão presente na própria janela de edição).

3. A janela de desmontagem (*disassembly*) será automaticamente realçada e o programa será carregado na memória do processador, 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.

   P1. Há outras pseudo-instruções nesse código? Quais?

4. Execute o programa (opção `Continue` ou `F3`). A mensagem `Running` aparecerá no topo da janela (em verde) indicando que o programa está sendo executado. Para parar a simulação, escolha `Stop` (F4). 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 simulador.

5. Retorne ao início do programa clicando no botão `Restart`. Agora use a opção `Step Into` (F2) para executar instrução por instrução. Observe como cada instrução altera o conteúdo dos registradores do processador (aba à esquerda).

6. Coloque o valor do contador de programa (`PC`) para 0 clicando neste registrador e digitando o valor 0. Note que esta ação tem o mesmo efeito que o botão `Restart`.

7. Desta vez adicione um *breakpoint* no endereço 0x24 (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.

   P2. Por que o breakpoint foi colocado neste endereço? Qual a importância do breakpoint?

8. Remova o breakpoint (clicando sobre ele). Agora, faça o contador de programa ficar com o valor 0x8, o que desconsidera as primeiras duas instruções que carregam o endereço de LIST para o registrador *r4*. Faça, também, com que o valor do registrador *r4* se torne 0x504. Execute o programa novamente.

   P3. Qual é o resultado da execução? Explique o que aconteceu.

P4. O uso da diretiva `.org` no código da Figura 1 é necessário? Qual sua função?
P5. Existe um erro sutil no código da Figura 1. Qual é esse erro? O que você faria para consertá-lo?

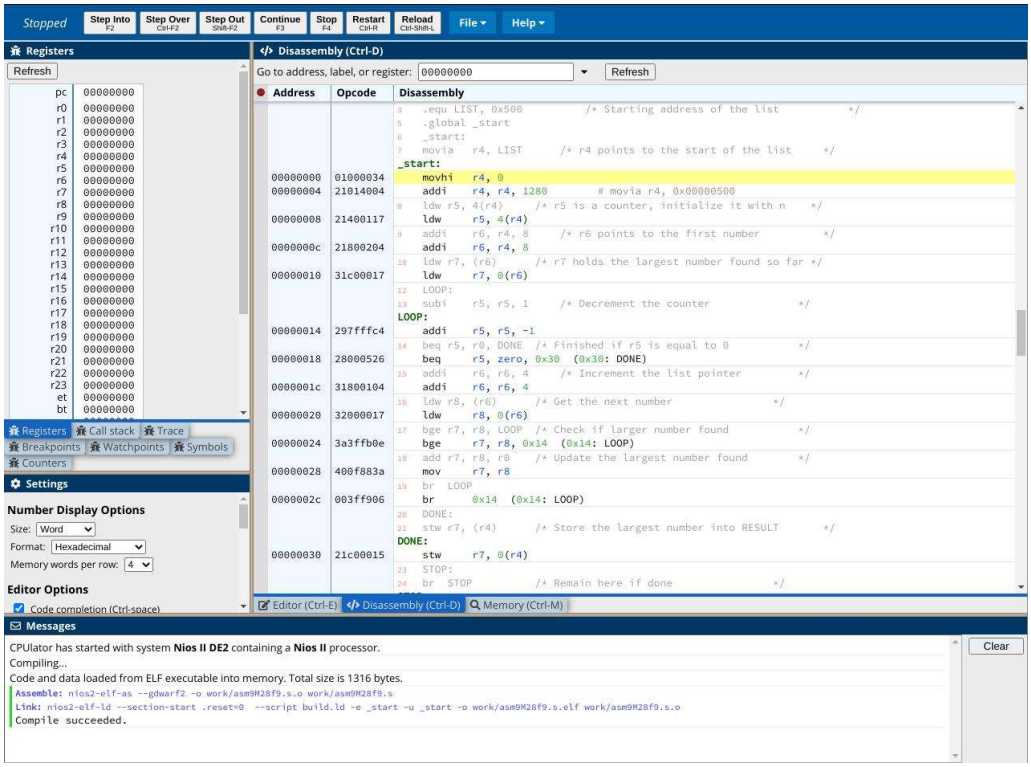

Figura 2: A visão do programa da Figura 1 com código desmontado.

Parte II

Nesta parte você deve escrever um programa em linguagem de montagem do Nios II para gerar os primeiros n números da série de Fibonacci (iterativamente). Nesta série, os primeiros dois números são 0 e 1, e cada número subsequente é gerado adicionando-se os dois últimos números da série. Por exemplo, para $n = 8$, a série é

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

Seu programa deve armazenar os números em posições sucessivas de memória, começando no endereço 0x1000. Coloque o valor de teste n na posição 0xffc.

Prossiga como a seguir:

1. Escreva um programa em linguagem de montagem do Nios II que compute a série de Fibonacci requisitada. Não esqueça dos comentários e de salvar o arquivo fonte. **Dica: escreva o pseudo-código antes para te auxiliar.**

2. Execute o programa para diferentes valores de n. Examine as posições de memória a partir de 0x1000 para verificar que seu programa está correto.