SIMULAÇÃO VHDL DO PROCESSADOR MRstd (*MIPS reduzido standard*)

Fernando Gehm Moraes
Ney Laert Vilar Calazans
18/novembro/2009

1. Salve em um diretório os seguintes arquivos disponíveis na página de conteúdos da disciplina:
   a) **mips_multiciclo.vhd**
   b) **mrStd_tb.vhd**
   c) **rotinas_aninhadas.asm**
2. A **montagem de código** pode ser feita automaticamente pelo simulador MARS. Abrir no simulador MARS o arquivo "rotinas_aninhadas.asm" e teclar F3. A seguir na opção de menu "File" do simulador selecione a opção "Memory Dump", salvando os segmentos de texto e dados como já discutido em aula (usem "Dump Format" selecionando "Text/Data Segment Window" e salve-os como, por exemplo, prog.log e data.log, respectivamente). Após isto, feche o simulador MARS.
3. **Adaptação do código objeto** para leitura pelo simulador VHDL – Comece concatenando os dois arquivos em um único, denominado "rotinas.txt". Abra o arquivo "rotinas.txt", e o modifique-o da seguinte forma:
   - Elimine o cabeçalho do arquivo texto gerado e acrescente no seu lugar uma linha antes da primeira linha contendo o código objeto do programa. A nova linha deve conter exatamente o texto **Text Segment,** iniciando na primeira coluna.
   - DA mesma forma, substitua a linha imediatamente antes da área de dados por uma linha contendo apenas **Data Segment,** iniciando na primeira coluna.
   - Deixe apenas as linhas do segmento de dados onde existem dados especificados no programa. Salve o arquivo "rotinas.txt" assim modificado. No exemplo em questão isto consiste em deixar no arquivo apenas a seguinte linha da área de dados:

```
0x10010000 0x00aa0000  0x0000bb00  0x000000cc  0x00000000
```

   - No final, o conteúdo do arquivo deve corresponder a (mostra-se abaixo apenas o início e o fim do novo arquivo rotinas.txt):

```
Text Segment
0x00400000  0x3c011001  lui $1,4097          24          la      $s1, var_a
0x00400004  0x34310004  ori $17,$1,4
0x00400008  0x8e310000  lw $17,0($17)        25          lw      $s1, 0($s1)
....
0x004000c0  0x01f87821  addu $15,$15,$24     92          addu    $t7,$t7,$t8
0x004000c4  0xafaf0004  sw $15,4($29)        93          sw      $t7,4($sp)
0x004000c8  0x03e00008  jr $31               94          jr      $ra

Data Segment
0x10010000  0x00aa0000  0x0000bb00  0x000000cc  0x00000000
```

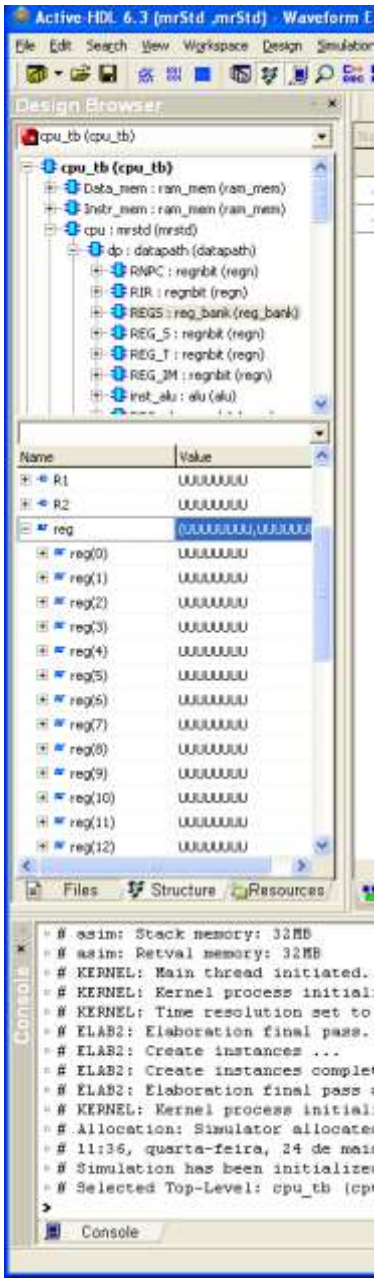
   - Lembre-se que as linhas contendo **Text Segment** e **Data Segment** foram inseridas manualmente no arquivo
   - De `0x00400000` até `0x004000c8` estará o código objeto do programa.
   - De `0x10010000` até o final da área de dados tem-se o código objeto da área de dados do programa.

4. Abra o simulador ActiveHDL e crie um novo projeto, tendo arquivos fonte os seguintes:
   - **mips_multiciclo.vhd**
   - **mrStd_tb.vhd**
   - **rotinas.txt**

5. Compilar os fontes usando a opção "Compila All with File Reorder" e inicializar a simulação (Não omitam este passo, opção de menu é "Simulation → Initialize Simulation"). Ao simular, devem ser observados os seguintes sinais:

– Em um janela *waveform* inserir inicialmente os sinais *ck*, *i*, e PS (todos parte dos sinais do Bloco de Controle do processador) e um conjunto de registradores do banco de registradores. Abaixo, mostra-se como fazer a seleção dos sinais.

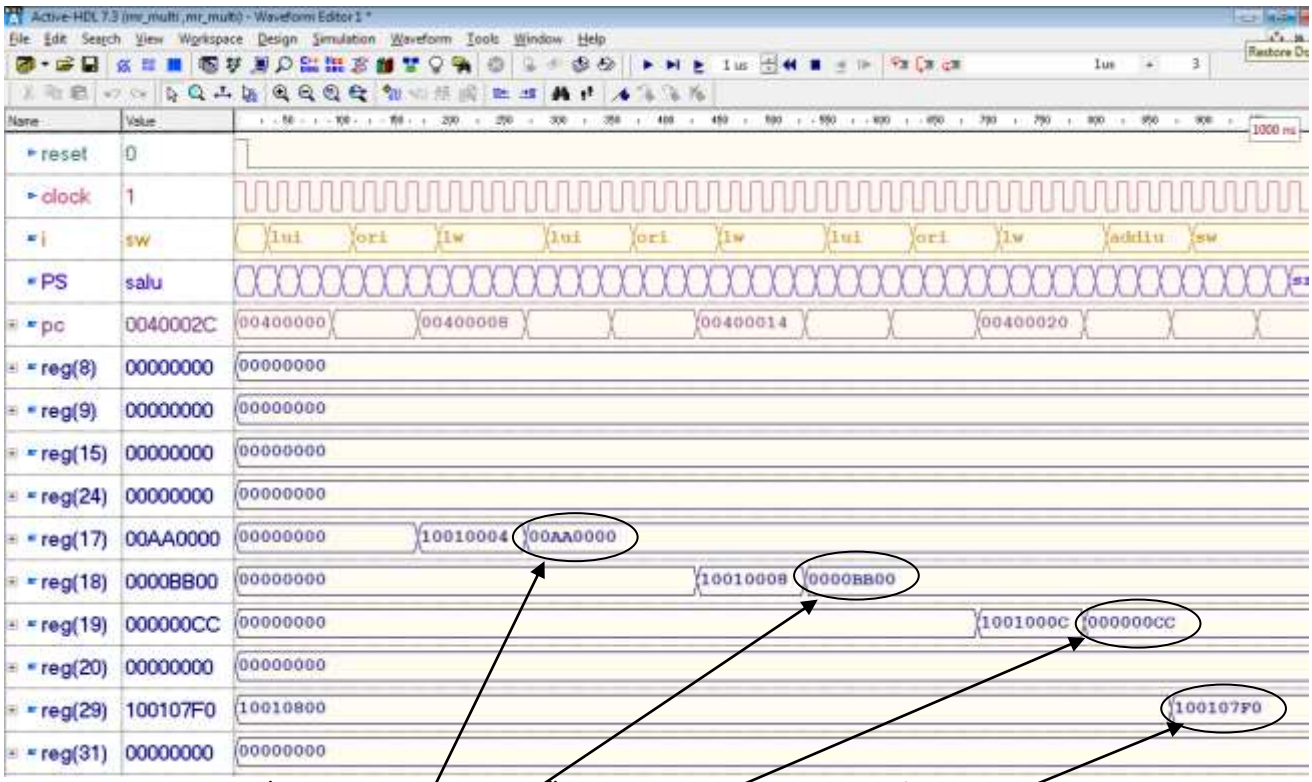
(a) Seleção dos Sinais contidos no bloco de controle – ck, i, PS.



(a) Seleção dos Sinais contidos no banco de registradores.

Selecionar os seguintes registradores:
$t0, $t1,$t7, $t8 = R8, R9, R15, R24
$s1, $s2, $s3,$4 = R17, R18, R19, R20
$sp, $ra = R29, R31

– **Simule o projeto por 1 us.** A janela abaixo ilustra a forma de onda que deve resultar da simulação.




```
la      $s1, var_a      # inicializa as variáveis
lw      $s1, 0($s1)
la      $s2, var_b
lw      $s2, 0($s2)
la      $s3, var_c
lw      $s3, 0($s3)

addiu   $sp,$sp,-16     # aloca 4 posições na pilha
```

6. As 4 instruções seguintes correspondem à escrita na pilha, memória de dados:

```
sw      $ra, 0($sp)
sw      $s1, 4($sp)
sw      $s2, 8($sp)
sw      $s3,12($sp)     # empilha o endereço ....
```

- Adicione a memória de dados na simulação (como se faz isto??)
- Re-simular o sistema por 1200ns
- Observar os endereços 7FC-7FF (em decimal, 2044 a 2047). Que valor está armazenado nestas posições? O que significa este dado neste lugar? Lembre-se que os dados são armazenados por um processador operando como ***little endian***!

7. Reinicialize a simulação e execute até 4.83 us, ou seja, até o final do programa. Comprove que se está no fim do programa a partir do valor do registrador PC (com quanto ele deve estar ao final do programa? Como "termina" este programa?)
   - Isto ocorre porque ao final da execução da simulação há um salto **`end: j end`**. Ou seja, o fica em laço eterno executando esta última instrução.

8. TAREFA: modifique o programa de tal forma que o resultado seja escrito em uma posição de memória, e mostre esta posição de memória com o valor lá armazenado (**`00AABBCC`**). O valor desta soma está também no registrador $s4 (registrador 20), o qual é escrito em 4650 ns. Usar um la e um sw, adicionados ao final do código.

RESPONDER

9. Algumas instruções do processador poderiam gravar o resultado da operação da ULA diretamente no banco de registradores, executando-as em 3 ciclos ao invés de 4. Pede-se:

a) Cite quais instruções poderiam ser executadas em 3 ciclos.
b) Qual a possível conseqüência desta alteração na duração do ciclo de *clock*?
c) Como o bloco de dados poderia ser modificado para atender a esta modificação?

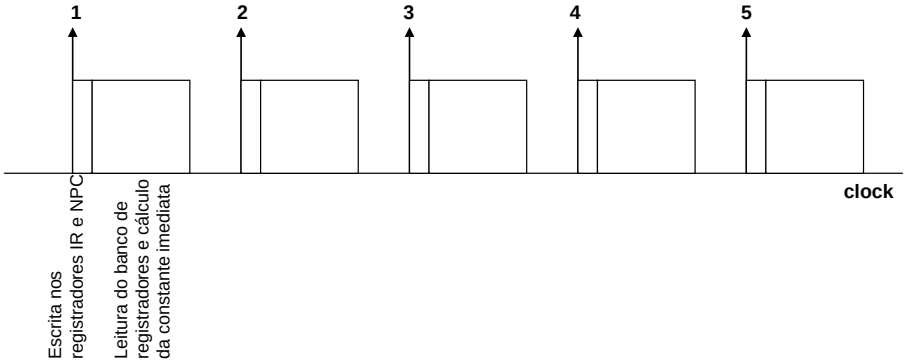
10. Um projetista resolveu economizar hardware, e moveu o comparador utilizado nos saltos (BEQ, BGEZ, BLEZ, BNE) para a ULA. Que conseqüências haveriam? Os saltos seriam realizados? Explique.

11. Que modificações seriam necessárias no processador multi-ciclo para operar com memória unificada (modelo *Von Neumann)*? Há aumento/redução no número de ciclos de relógio para a execução das instruções?

12. Considerando apenas os tempos de acesso às memórias, escrita nos registradores, banco de registradores e ULA, supor os seguintes tempos:

- Memória de instruções: gasta 2ns para leitura.
- Memória de dados: gasta 2ns para leitura ou escrita.
- Banco de registradores: gasta 2ns para leitura ou escrita.
- ULA: gasta 4ns para executar.
- Tempo de escrita nos registradores: gasta 1ns.

a) Qual o período mínimo de relógio para a implementação multi-ciclo? Diga quais registradores são escritos e o "trabalho" executado no ciclo. O primeiro ciclo já está detalhado.



b) Qual o período mínimo de relógio para a implementação monociclo?

c) Qual o estágio que realiza a menor quantidade de trabalho? Este estágio poderia ser removido? Qual o benefício em termos de desempenho?