1o Trabalho Prático de Sistemas Operacionais I - 2015

Desenvolva uma aplicação concorrente em C/PThreads no Linux, a qual simule o comportamento da CPU MIPS multiciclo [1]. Esta CPU deve executar estas nove instruções MIPS: ***add, sub, and, or, slt (set-on-less-than), lw (load word), sw (store word), beq*** (***branch-on-equal*** - desvio condicional) e ***j*** (***jump*** – desvio incondicional) [1]. Use como base o caminho de dados completo (vide figura abaixo) e os sinais de controle da CPU MIPS multiciclo, ambos descritos em [1].

Cada unidade funcional descrita no diagrama de blocos para a CPU multiciclo deve ser uma ***thread*** que executará concorrentemente (isso inclui cada multiplexador, os deslocadores à esquerda e a escrita nos registradores ***PC***, ***IR***, **MDR**, ***A***, ***B*** e ***ALUOut***). As portas ***E*** e ***OU***, usadas para permitir a escrita em ***PC***, também formam uma ***thread*** à parte. Essas ***threads*** executam independentemente e devem comunicar/sincronizar corretamente para garantir a semântica da execução da CPU MIPS multiciclo.

Os registradores ***PC***, ***IR***, ***MDR***, ***A***, ***B***, ***ALUOut*** e do banco de registradores são variáveis globais em memória e são compartilhados entre as **threads**. A memória RAM também é global.

O sinal de *clock* deve ser emitido (simulado) pela ***thread main***, isto é, ele marca o início de um novo ciclo para todas as outras ***threads*** e também permite a escrita (de fato) nos elementos de lógica sequencial. Considere que as informações lidas dos elementos de lógica sequencial em um ciclo são aquelas informações escritas no ciclo anterior (não se pode ler uma informação que foi gravada neste mesmo ciclo).

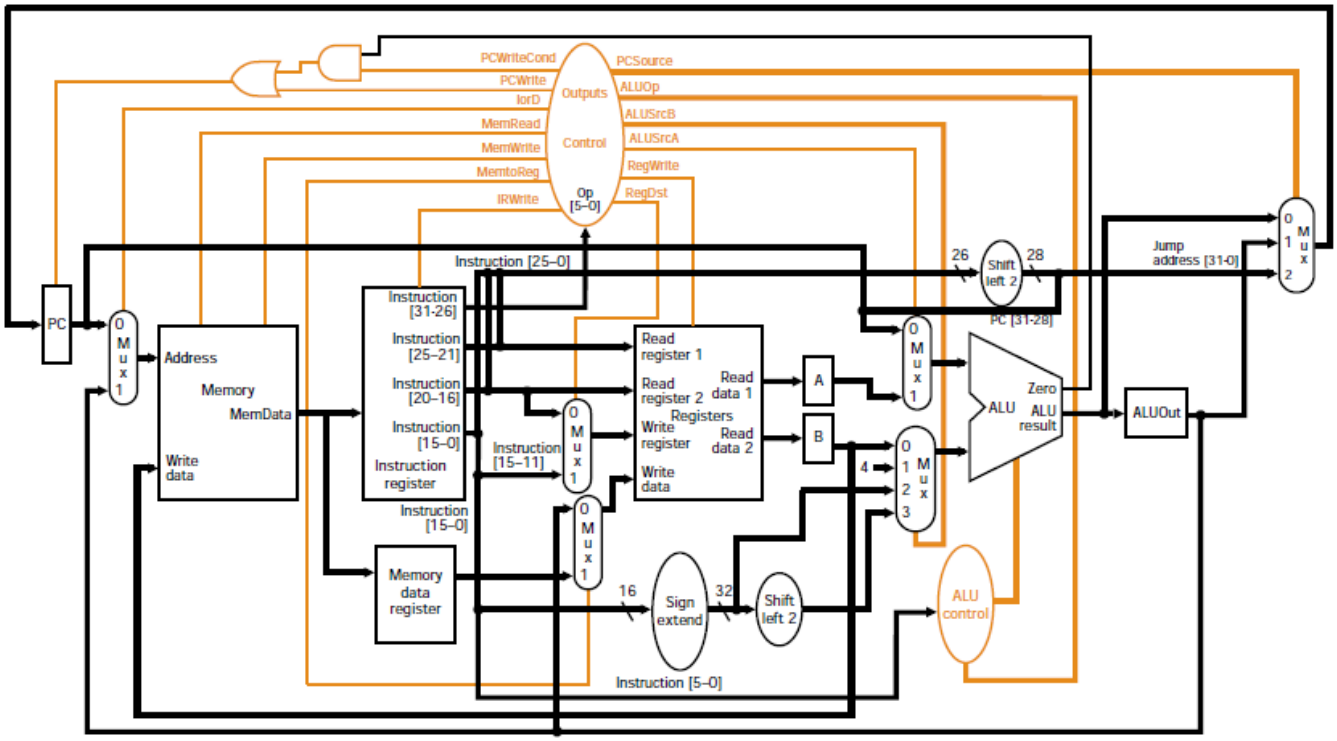
As unidades funcionais executam em todos os ciclos. Cabe apenas aos sinais de controle emitidos pela UC impedir que algo "errado" seja feito e/ou considerado. A geração dos sinais de controle pela UC é o primeiro ponto que requer sincronização após o ciclo ter começado. Outros pontos de sincronização ocorrem em algumas das entradas dos multiplexadores. Em outras palavras, observe o fluxo de execução do ciclo de instrução para garantir que as dependências serão respeitadas, caso contrário não será atingida a semântica correta da CPU simulada. O início de um novo ciclo só pode ocorrer após se atingir toda a funcionalidade esperada para o ciclo atual.

Desenvolva a aplicação concorrente segundo as diretrizes passadas aqui e em aula. As questões omissas e/ou ambíguas serão fixadas pelo professor. Qualquer dúvida, entre em contato com o professor para orientação.

O trabalho será entregue via *moodle*, um por grupo. IMPORTANTE: destaque como comentário no início do código o número do grupo e os nomes dos integrantes do grupo que de fato participaram do desenvolvimento do trabalho. Os nomes que não aparecerem nessa relação ficarão com nota "zero". A correção considerará a funcionalidade e a qualidade do programa desenvolvido.

O seu código deverá ser capaz de executar corretamente um programa simples inserido na memória, contendo as nove instruções solicitadas. A saída do programa deve ser apenas: (1) o conteúdo das posições de memória RAM que foram alteradas pelo programa e (2) o conteúdo dos registradores. O trabalho será apresentado pelo grupo todo ao professor.

Diagrama de blocos da CPU MIPS multiciclo a ser considerada neste trabalho prático [1].



Referências

[1] Patterson, D.A., Hennessy, J.L. Computer organization and design : the hardware/software interface, 3ª ed., Elsevier, Amsterdam, 2005.