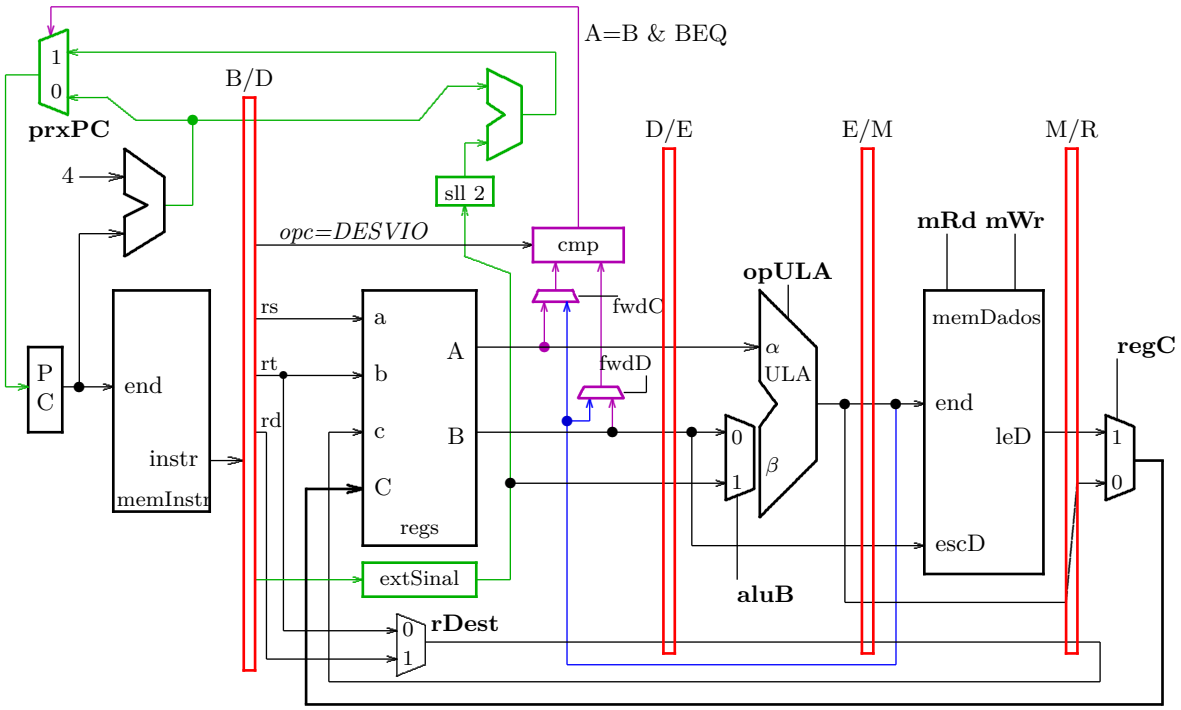
UFPR – Bacharelado em Ciência da Computação
CI212 – Organização e Arquitetura de Computadores – 2016-2

Primeira Prova

1. Você se refugiou no número 12 do Largo Grimmauld para se preparar para o exame dos Níveis Ordinários de Computação. Monstro, seu elfo doméstico, leu o livro de Arquitetura enquanto você jogava xadrez, ao invés de estudar. Após longa pausa, você retorna preguiçosamente ao trabalho e Monstro dispara: "Vale a pena implementar a instrução JUMP THROUGH MEMORY no processador segmentado?". Por JUMP THROUGH MEMORY o elfo se referia a um salto para uma posição apontada por um endereço em memória:

`jtm desl(r1) # PC <- M[ R[r1] + extSinal(desl) ]` formato I

Para escapar à flagrante e evidente humilhação, você deve (i) mostrar como se implementa tal instrução; (ii) mostrar os sinais de controle que ficam ativos em cada estágio do processador; e (iii) responder se vale à pena, justificando esta resposta, sem esquecer que Monstro é um calhorda insuportável e exigente que nunca lhe dará sossego em caso de resposta errada. [10 pontos]



2. Não contente em desafiá-lo uma única vez, Monstro lhe propõe a tradução do trecho de código ao lado para *assembly* do MIPS, atendendo às convenções para a implementação de funções. Mais ainda, o elfo petulante exige que seu código execute corretamente no Processador Padrão do Ministério da Magia, com adiantamento e com *branch-delay slot*. Ele rosna que `X` e `Y` são adjacentes e `NN` ≤ 1024. [10 pontos]

```
int X[NN], Y[NN], pv;
...
pv = accio(X,Y,NN);
...
int accio(int *p, int *q, int n) {
   int i; int s = 0;
   for(i = 0; i < n; i++)
      s = s + p[i] * q[i];
   return s;
}
```

3. Você encontra Hagrid no pátio interno e ele lhe pede ajuda para semear um canteiro com abóboras gigantes. Você imediatamente se dá conta de que o canteiro tem a geometria de uma cache com capacidade para 32 palavras, associatividade quaternária, e blocos de duas palavras. Hagrid lhe pede para mostrar como ficaria o canteiro depois de semeado com sementes de números: 3, 11, 16, 24, 17, 12, 10, 5, 14, 23, 62. O que você faz quando todos os buracos de uma fileira do canteiro já estão tomados, e uma nova semente cairia num daqueles buracos? [10 pontos]

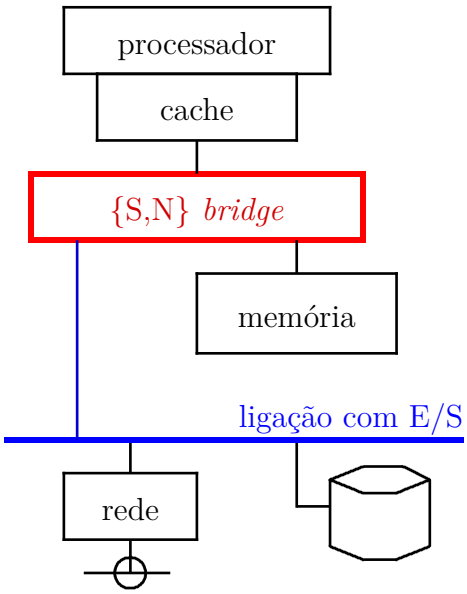
Segunda Prova

4. Você está escrevendo um módulo do sistema operacional que trata as operações de uma versão nova de um periférico. A versão antiga, em ferro fundido e que foi desenvolvida pelos anões há 100 anos, necessitava de 10^5 ciclos para responder a uma leitura. A nova versão, em mithril, necessita somente de 10^3 ciclos. Seu parceiro de jornada, Peregrin Tûk, muito preocupado com o desperdício de energia, sugere colocar o processador a hibernar enquanto se espera pela interrupção que avisa que uma operação do periférico completou.

Você concorda ou discorda de Pippin? Justifique sua resposta como se a sobrevivência de Gondor dependesse do sucesso do seu projeto. [10 pontos]

5. Gandalf lhe pede ajuda para projetar uma caixa para acomodar traduções da lingua comum para todos os outros idiomas da Terra Média. Enquanto fala da encomeda, ele insiste em lhe mostrar a mão aberta, com os cinco dedos magros e longos estendidos. Após pensar uns instantes, você se dá conta de que ele precisa de uma TLB com 2^5 elementos e endereços de 2^5 bits. Gandalf, em sendo cuidadoso, lhe pede um diagrama detalhado do projeto, que mostre todas as contas, e justifique *quaisquer* suposições necessárias para completar o projeto da sua caixa tradutora. [10 pontos]

6. Galadriel lhe chama até o espelho e lhe mostra o diagrama ao lado.
Sem que ela emita uma sílaba, você sabe que deve:
(a) desenhar um diagrama detalhado do mecanismo para transferir dados entre memória e periféricos; e
(b) escrever, em pseudo-código, a rotina que permite transferir 64Kbytes da memória para a rede.
[10 pontos]



Exame Final

7. Você deve acrescentar ao processador o circuito que suporta uma nova instrução que executa a multiplicação dos conteúdos de dois registradores e soma o produto ao conteúdo do terceiro registrador. O resultado é armazenado nos registradores `hi` e `lo`. Esta nova instrução é chamada de *multiply-add*, `madd`, com resultado em 64 bits.

MULTIPLY-ADD: `madd rs,rt,rd  # hi&lo ← (R[rs]*R[rt]) + extSinal64(R[rd])`

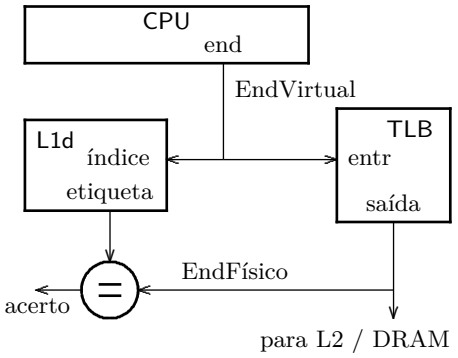
A nova instrução tem formato R.
Por limitações da tecnologia de implementação, o circuito que efetua a operação deve ser implementado em dois estágios, com a multiplicação no estágio EXEC e a soma no estágio MEM.

```
#define i64 (long long)
i64 X[2048];
int A[2048],B[2048],C[2048];

for(i=0; i<2048; i++)
   X[i]=(i64)( (A[i]*B[i])+C[i] );
```

Mostre como acrescentar a instrução `madd` ao conjunto de instruções do processador. Sua resposta consiste de quatro partes:
(i) um diagrama de blocos claro e limpo mostrando os componentes e as ligações internas da nova unidade funcional *multiply-add*; [5 pontos]
(ii) um desenho claro e limpo, indicando os circuitos e ligações adicionais para interligar a nova unidade funcional ao processador; [10 pontos]
(iii) uma definição clara e precisa das condições de bloqueio para evitar que os resultados de `madd` sejam usados incorretamente – bloqueio não é o mesmo que adiantamento; [10 pontos]
(iv) traduza o programa acima para assembly do MIPS; seu programa *deve* usar a instrução `madd`. *Para facilitar a correção indique os registradores como* `ra`, `rb`, *etc. Use uma folha inteira para escrever o programa em assembly. O diagrama do processador está no verso.* [25 pontos]

8. Esta questão tem dois itens. A figura ao lado mostra o circuito de endereços de um sistema com memória virtual com cache primária de dados, cache secundária e memória dinâmica (DRAM).



(a) Descreva a operação do primeiro nível da hierarquia de memória, usando como exemplo um load e um store; [10 pontos]

(b) justifique cuidadosamente este projeto do ponto de vista do Sistema Operacional. [10 pontos]

9. Considere um multiprocessador simétrico com 4 processadores interligados por um barramento. As caches são mantidas coerentes por um protocolo de invalidação, que é mostrado na tabela. Os blocos das caches são de 8 palavras de 32 bits. O protocolo usa escrita forçada na primeira escrita em um bloco; após a primeira escrita, o protocolo usa escrita preguiçosa. Quando um bloco MODificado é substituído na cache, todo o bloco deve ser copiado para a linha em memória. Este protocolo usa os estados EXCLusivo e MODificado para reduzir o tráfego causado por escritas.

A tabela contém uma representação do diagrama de estados e inclui as ações dos controladores das caches. O topo da tabela indica os eventos e a tabela mostra o estado final e a ação do(s) controlador(es) da(s) cache(s) envolvidas. *Snoop*s são operações iniciadas por caches remotas.

estado	RD hit	RD miss	WR hit	WR miss	RD snoop	WR snoop
mod	mod	sha/wrBack	mod	excl/wrBack	sha/wrBack	inv/wrBack
excl	excl	sha	mod	excl	sha	inv
sha	sha	sha	excl/purge	excl/fill	sha	inv
inv	X	sha/fill	X	excl/fill	X	X

wrBack — única cópia de bloco no estado MOD deve ser gravada em memória. Durante atualização, outras caches podem copiar novo valor do barramento;

fill — bloco deve ser preenchido com valor fornecido pelo barramento, que pode ser da memória ou de outra cache;

purge — todas as cópias devem ser invalidadas pelos outros controladores de caches. A cache que emitiu comando mantem cópia exclusiva;

SHAred — cópia válida, só de leitura, compartilhada com outras caches. Antes de atualização, outras cópias devem ser invalidadas;

EXCLusiva — única cópia, memória mantem cópia atualizada. Nova escrita torna MODificada;

MODificada — cópia exclusiva, diferente da memória. Quando for reposta por outro bloco, memória deve ser atualizada (wrBack).

Qual o estado final das caches, supondo o estado inicial mostrado abaixo, e a execução dos seguintes comandos pelos três processadores. Os comandos estão mostrados na ordem absoluta de tempo; os índices dos blocos nas caches são irrelevantes, e as caches são infinitas. [30 pontos]

P0: MOD | v[0]..v[7] **P1**: SHA | u[0]..u[7] **P2**: SHA | u[0]..u[7]

P0

```
1  ...
2  ...
3  ...
4  ...
5  ...
6  a=0;
7  for(j=0; j<8; j++)
8    a += t[j] + v[j+4];
9  ...
10 ...
11 ...
12 ...
13 ...
14 ...
15 for(j=0; j<16; j++)
16   u[j+8] = t[j+4]*v[j];
17 ...
18 ...
```

P1

```
1  ...
2  ...
3  ...
4  u[5] = 0;
5  ...
6  ...
7  ...
8  ...
9  ...
10 u[6] = 0;
11 ...
12 ...
13 ...
14 ...
15 ...
16 ...
17 ...
18 u[7] = 0;
```

P2

```
1  for(i=0; i<7; i++)
2    t[i] = u[i]*v[i];
3  ...
4  ...
5  ...
6  ...
7  ...
8  ...
9  ...
10 ...
11 ...
12 for(i=0; i<17; i++)
13   t[i] = u[i]*v[i];
14 ...
15 ...
16 ...
17 ...
18 ...
```