µProcessador 6 Memória de Dados

Incluir no circuito uma memória RAM e instruções para usá-la. O modo mais simples de realizar isso é seguir o modelo do MIPS, com instruções *lw* e *sw* e arquitetura Harvard.

O requisito que deve ser obedecido é que o cálculo do endereço deve usar uma variável (um registrador, para loops sobre vetores, por exemplo), podendo ou não incluir uma constante de deslocamento como no MIPS.

Abaixo um exemplo com instruções que não incluem constantes:

```
li    $3,0x1E     # endereço carregado em $3
lw    $1,($3)     # dado carregado em $1
```

Detalhes de Implementação

As RAMs disponíveis são a RAM: 1-PORT acessível através do botão do MegaWizard do *Symbol Tool* e recomendada pela Altera e outras que estão em obsolescência, como a LPM_RAM_DQ (podem usar) e outras mais complicadas altdpram e altsyncram.

Alternativas de Instruções

- Usar instruções CISC diferentes do paradigma *load/store* como, por exemplo, instruções que somam registros com conteúdos de memória.
  Neste caso, poderíamos ter: `addm $1,$1,($3)` para somar o registrador $1 com o valor do endereço de memória apontado por $3 ou `addmi $1,$1,(0x1C)` para somar com o conteúdo do endereço 0x1C, ou `addm $1,$0,($3)` para ler do endereço $3 para o reg. $1).
- Acrescentar um registrador especial fora do banco (digamos $m, bem CISC) que guarda o endereço que será acessado pelas instruções de memória:
  ```
  ldm $m,0x1C # sempre obrigatório com $m
  lw  $1,$m   # sempre obrigatório com $m
  ```
- Usar instruções com um registrador fixo (p. ex.: o $3 pode ser usado em qualquer operação além de *lw/sw*, mas nestas ele obrigatoriamente contém o endereço de memória acessado).

Entrega Eletrônica

O prazo é o habitual duas semanas até 13h00 (atenção nesse).

Observe no plano de aulas a data final para entregar pronto e funcionando o programa que preenche a memória com os números primos, que devem ser calculados pelo programa de acordo com as limitações (largura dos registradores, tamanho da memória etc). Depois disso, um loop deve enviar os primos calculados para o registrador de debug. *É obrigatório entregar, junto à apresentação final, o código em assembly que calcula os primos e os opcodes respectivos.*

Dois arquivos texto com códigos hexa devem ser entregues:

- **"checksum.txt":** é um *loop* que lê seis endereços de memória, calcula a soma destes dados e escreve esta soma no endereço subsequente, mostrando-a no simulador.
- **"ram.txt":** contém trechos de códigos, separados entre eles com uma linha em branco, com as seguintes funções:
  1. lê o conteúdo do endereço 02h para o registrador 1
  2. escreve o valor do registrador 1 no endereço 02h
  3. troca os valores dos registradores 1 e 3
  4. lê para o registrador 1 o conteúdo do endereço de memória dado pelo registrador 1

Favor enviar novamente os arquivos .txt com códigos hexa das práticas anteriores. Caso haja alterações da codificação em relação às práticas anteriores, lembre-se de **atualizar estes arquivos** e testá-los.

Checklist de Entrega

Circuitaria:

- ✔ Não há pinos de entrada deixados em aberto (i.e., desconectados)
- ✔ Todos os flip-flops/contadores possuem pino de RESET
- ✔ Não há acentuação ou espaços nos componentes
- ✔ Não há acentuação ou espaços nos nomes dos arquivos nem no caminho das pastas

Arquivos anexados no email:

- ✔ O arquivo de projeto compactado **.qar** *atualizado*
- ✔ O **projeto.txt** com as configurações *atualizadas*
- ✔ O **checksum.txt** e o **ram.txt** como descrito anteriormente
- ✔ Os códigos de ROM para realizar o cálculo pedido, **checksum.mif**
- ✔ Os testes **.vwf** que vocês utilizaram, para me ajudar caso preciso
- ✔ Os outros arquivos **.mif** com os programas testados nos .vwf