Projeto de conclusão da disciplina "Arquitetura de Computadores" (PCS-2405) - 2016

Implementação e caracterização do processador "FemtoMIPS".

Introdução

O propósito deste projeto é a implementação, através da codificação da descrição comportamental com uso da linguagem VHDL, o processador FemtoMIPS. Este processador é uma versão reduzida e simplificada da família de processadores MIPS. Este projeto deve ser realizado em grupos de trabalho até 4 alunos, sendo altamente recomendada a formação de grupos de trabalho de ao menos 3 alunos.

Além da implementação, é o objetivo deste projeto a caracterização do desempenho deste processador, através da produção de relatórios detalhados de execução de programas selecionados para este fim. Para este fim, espera-se que a implementação descrita em VHDL seja fidedigna no que tange a todos os tempos de resposta de cada componente do Fluxo de Dados (FD) e Unidade de Controle (UC), além de prover contadores e sensores para todas as métricas relevantes. Este documento se divide em três partes. Na primeira parte, é descrita a Arquitetura do Conjunto de Instruções (ACI) do FemtoMIPS, a qual é uma versão simplificada da ACI MIPS32. Na segunda parte, são descritas a estrutura geral da MicroArquitetura (MA), bem como as especificações (i.e., condições de contorno) para a sua implementação.

Finalmente, a terceira parte descreve caracterização em termos de suas componentes fundamentais, as suas métricas, suas cargas de trabalho e informações relevantes para a produção do relatório de caracterização do desempenho do processador.

1 – Arquitetura do Conjunto de Instruções (ACI).

1.1 – Conjunto de operações

1.1.1 – Instruções de Acesso à Memória

- *lw: Escreve o conteúdo de uma posição da Memória de Dados (MD) em um registrador.*
- *sw: Escreve o conteúdo de um registrador em um posição de memória.*

1.1.2 – Instruções Lógico-Aritméticas

- *add: Soma dois operandos inteiros com sinal*
- *addi: Soma um valor fixo inteiro a um operando inteiro*
- *addu: Soma dois operandos inteiros sem sinal*
- *slt: Compara dois operandos, e assume valor "um" se um deles (sempre o mesmo) é menor que outro*
- *slti*: *Compara dois operandos, um em registrador e outro imediato com sinal estendido, e assume valor "um" se o conteúdo do registrador for menor que o imediato.*
- *sll*: *Deslocamento lógico à esquerda de um registrador pelo número de bits especificados no campo shamt da instrução.*

1.1.3 – Instruções de desvio

- *beq: Compara o valor de dois operandos, e desvia o fluxo de execução para um endereço especificado na Memória de Instruções (MI) caso estes sejam iguais*
- *bne*: *Compara o valor de dois operandos e desvia o fluxo de execução para um endereço especificado na MI caso estes sejam iguais.*
- *j: Desvia o fluxo de execução para um endereço especificado na MI*
- *jal: Guarda o endereço da próxima instrução em um registrador específico (número 31) e desvia o fluxo de execução para um endereço especificado na MI.*
- *jr*: *Desvia o fluxo de execução para um endereço na MI especificado pelo conteúdo de um registrador.*

1.2 – Armazenamento Interno

- *Operandos são referenciados de forma explícita, pela indicação de dois registradores que contém os valores daqueles.*

- *Operações com 3 endereços (0 de MD) e 3 operandos.*
- *Tipicamente designado "Registrador/Registrador" ou "Load/Store".*

1.3 – Codificação e Representação

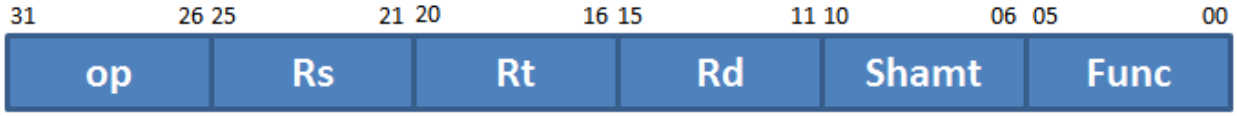
- *Representação de números inteiros em 32 bits (em complemento de 2)*
- *Representação de números de ponto flutuante em 64 bits (IEEE 754)*
- *Espaço de Endereçamento de Memória: 32 bits. Embora o espaço lógico de memória possua 32 bits de endereço, neste projetos deve ser implementado um espaço com 16 bits de endereço, ou seja, somente 64 Kbytes.*
- *Menor unidade endereçável na Memória: 8 bits*
- *Acessos à memória alinhados.*

1.3.1 – Instruções

- *Tamanho fixo de 32 bits, com os seguintes campos:*
  - *código de operação (op): identifica uma instrução específica (6 bits)*
  - *registrador de origem (rs): primeiro operando (5 bits)*
  - *registrador alvo (rt): segundo operando (5 bits)*
  - *registrador de destino (rd): resultado de operações (5 bits)*
  - *magnitude de deslocamento (shamt): número de bits de deslocamento (8 bits)*
  - *código de função (funct): indica uma variante específica de determinada operação (6 bits)*

1.31.1 – Instruções Tipo R (e.g., lógico-aritméticas)

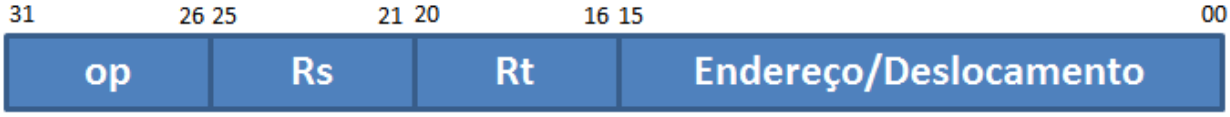
O diagrama abaixo representa a disposição de palavras de



- *instrução add*
  - *op=0*
  - *funct=32*
- *instrução slt*
  - *op=0*
  - *funct=42*
- *instrução jr*
  - *op=0*
  - *funct=08*
- *instrução addu*
  - *op=0*
  - *funct=33*

- *instrução sll*
  - *op=0*
  - *funct=00*

1.31.2 – Instruções Tipo I (e.g., acesso a memória)



- *instrução lw*
  - *op=35*
- *instrução sw*
  - *op=43*
- *instrução addi*
  - *op=8*
- *instrução beq*
  - *op=4*
- *instrução slti*
  - *op=10*

1.3.1.3 – Instruções Tipo J (e.g., salto incondicional)

- *instrução bne*
  - *op=5*
- *instrução j*
  - *op=2*
- *instrução jal*
  - *op=3*

1.3.2 – Modos de Endereçamento

- *Imediato (16 bits)*
- *Indireto*
- *Deslocamento*

1.4 – Estado do processador

1.4.1 – 32 registradores de propósito geral inteiros de 32 bits

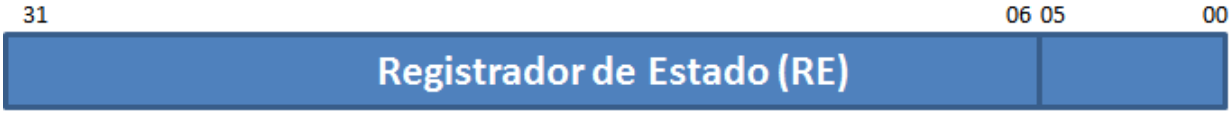
Registrador	*Nome*	*Função*	*Preservado*
0	*$zero*	*Constante 0*	*sim*
1	*$at*	*Reservado para Assembler*	*sim*
2-3	*$v0-$v1*	*Valores de retorno*	*não*
4-7	*$a0-$a3*	*Argumentos de procedimentos*	*não*
8-15	*$t0-#t7*	*Temporários*	*não*
16-23	*$s0-$s7*	*Preservados*	*sim*
24-25	*$t8-$t9*	*Temporários*	*não*
26-27	*$k0-$k1*	*Reservado para o SO*	*sim*
28	*$gp*	*Ponteiro Global*	*Sim*
29	*$sp*	*Ponteiro de Pilha*	*Sim*
30	*$fp*	*Ponteiro de Quadro*	*Sim*
31	*$ra*	*Endereço de Retorno*	*Sim*

Nesta versão do FEMTMIPS serão utilizados somente 8 registradores, de 0 – 7. O registrador para armazenamento de endereço de retorno ($ra) será o 7, o registrador $fp será o 6, o $sp será o 5 e o $gp será o 4. Os registradores de 0 à 3 são utilizados como de propósito geral.

1.4.2 – 4 registradores de propósito específico de 32 bits

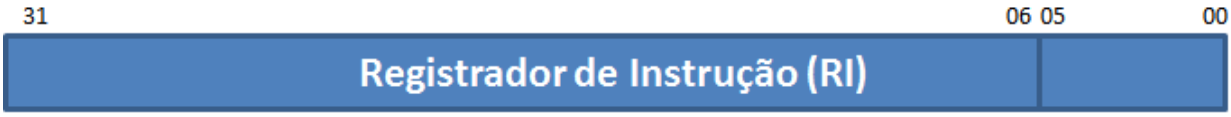
- *Contador de Instruções (CI) de 32 bits*

- *Registrador de Estado (RE) de 32 bits*



- *Registrador de Causa (RC) de 32 bits*

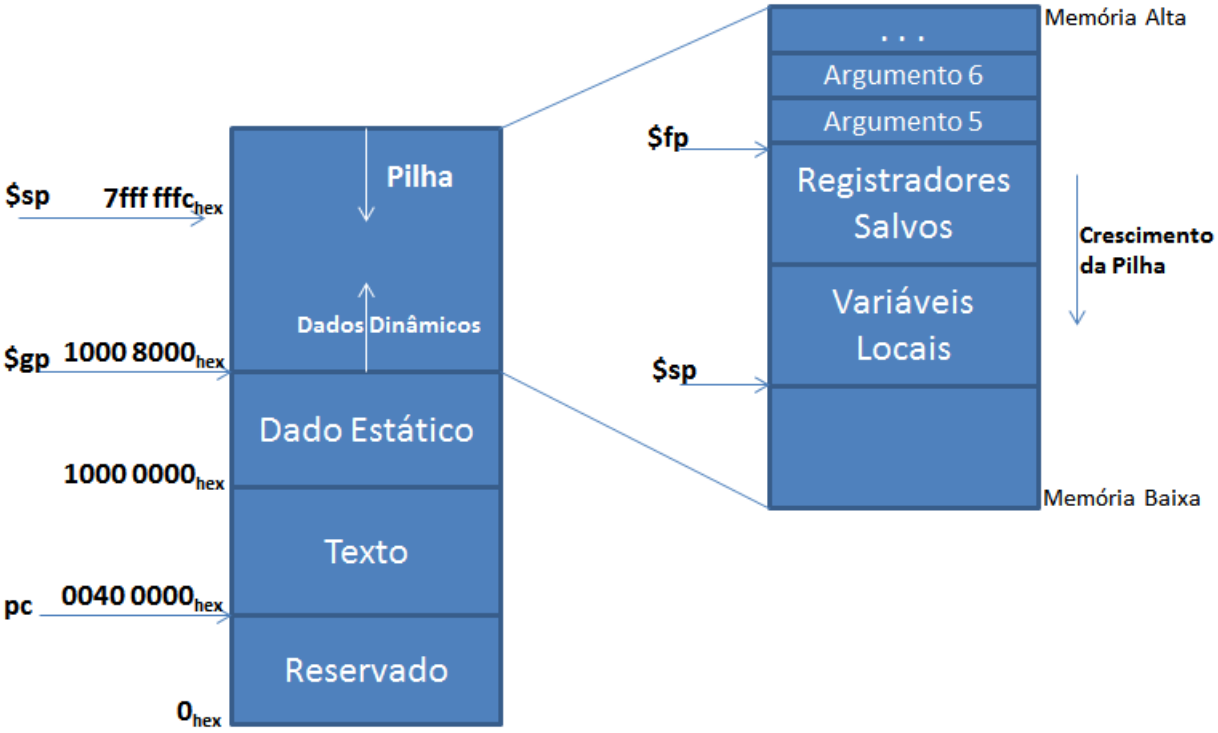
- *Registrador de Instrução em Exceção (EI) de 32 bits*



1.5 – *Modelo de Execução*

- *Chamada de procedimento simples.*
- *Chamada de procedimento recursivo.*

1.6 – *Subdivisão do Espaço de Memória*



Embora o mapeamento de endereços do MIPS seja o que se encontra mostrado na figura acima, neste projeto deve ser implementado o seguintes mapeamento:

***pc** = 0000_{hex};*
Início da área de dados estáticos = 0100_{hex}
***$gp** = 0200_{hex}*
***$sp** = $FFFF_{hex}$*

Os sentidos de crescimento dessas áreas de memoria são mantidos conforme esquema original do processador MIPS.

2 MicroArquitetura (MA).

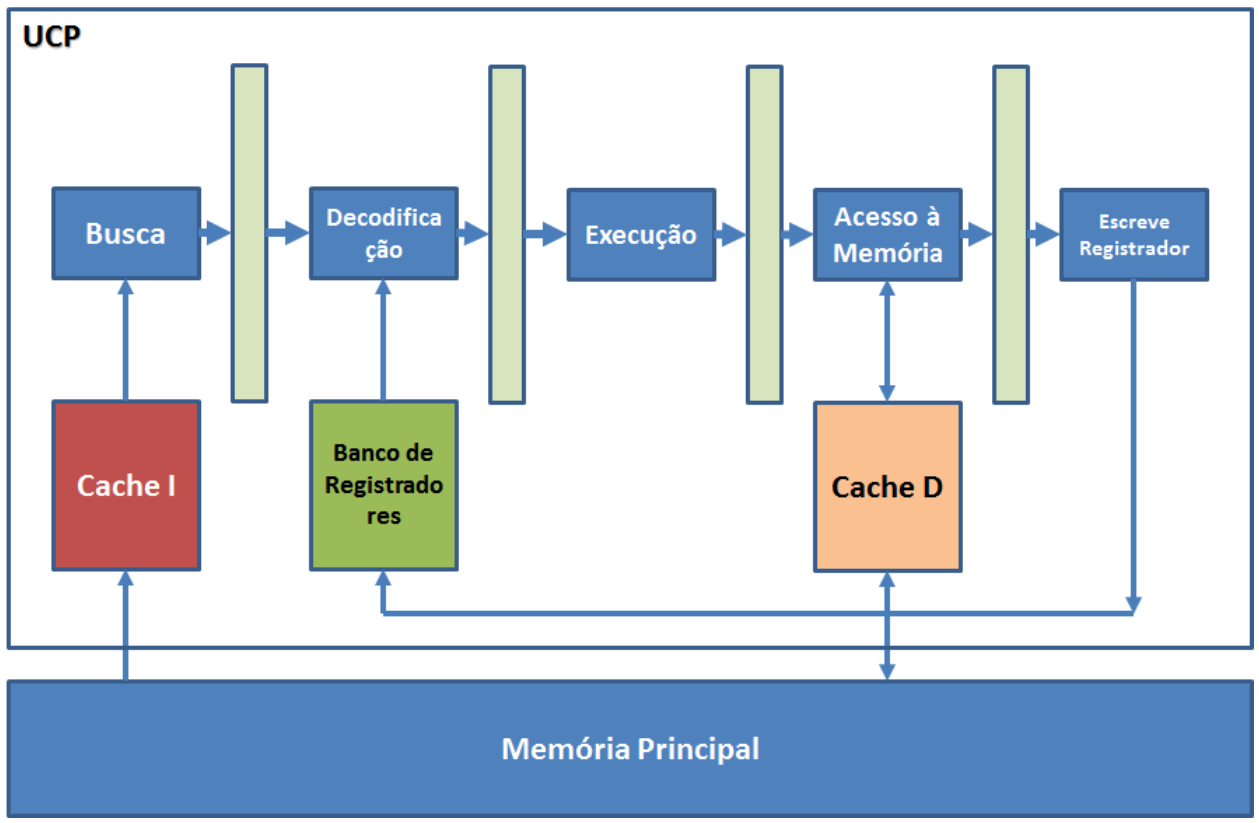
2.1 – Pipeline

A MA deste processador deve ser organizada como um pipeline de 5 estágios. A descrição de cada estágio é a que segue:

- ***Busca****: Conteúdo da memória cujo endereço é armazenado no contador de instruções é lido e o contador de instruções (pc) é atualizado.*

- ***Decodificação**: Parte deste conteúdo é enviado a Unidade de Controle (UC) do processador e parte para o banco de registradores, a fim de permitir que a UC determine quais são as operações e os operandos.*
- ***Execução**: As operações são executadas sobre os operandos através da alimentação destes em uma ULA controlada pela UC. No caso da instrução lw o endereço efetivo é calculado neste estágio.*
- ***Acesso a Memória**: Caso a instrução processada exija acesso à memória (lw ou sw), este é executado neste estágio (leitura no caso de lw e escrita no caso de sw), se a instrução sendo executado não necessita acesso à memória este estágio não executa nada. No caso da instrução de desvio, se ele tiver que ser realizado é neste estágio que o novo endereço é carregado no contador de instruções (pc).*
- ***Escreve de Volta**: O resultado da Execução ou do Acesso a Memória (lw) é armazenado no banco de registradores.*

O esquemático abaixo representa as principais interconexões entre cada estágio e a hierarquia de memória na MA especificada aqui.



Este pipeline deve observar as seguintes restrições e características operacionais:

- *O Tempo de Operação (TO) nos estágios Decodificação, Execução e Escreve de Volta é sempre inferior a 1 ciclo de relógio.*

- *O TO dos estágios Busca e Acesso a Memória dependerá das latências da hierarquia de memória, tratada a seguir.*
- *O Banco de Registradores (BR) deve ter uma latência de leitura e escrita menores que metade de um ciclo cada (i.e., deve ser possível ler e escrever o banco de registradores em um único ciclo).*
- *O BR opera de modo a garantir a prioridade da escrita sobre a leitura, de modo que uma requisição de escrita sempre "bloqueie" a requisição de leitura até que a primeira seja finalizada.*
- *O cálculo de endereço nas instruções de desvio condicional é realizado no estágio de Execução. O endereço efetivo é então propagado ao Contador de Instruções (pc), no estágio de Acesso a memória.*
- *O pipeline deve possuir uma "Hazard Detection Unit", que permita ao mesmo detectar dependências de Dados ou de Controle no fluxo de instruções executadas e tomar as ações apropriadas.*
- *Entre as ações apropriadas estão a inserção de bolhas de pipeline ou a propagação de valores de registradores através da "leitura antecipada" "forwarding".*

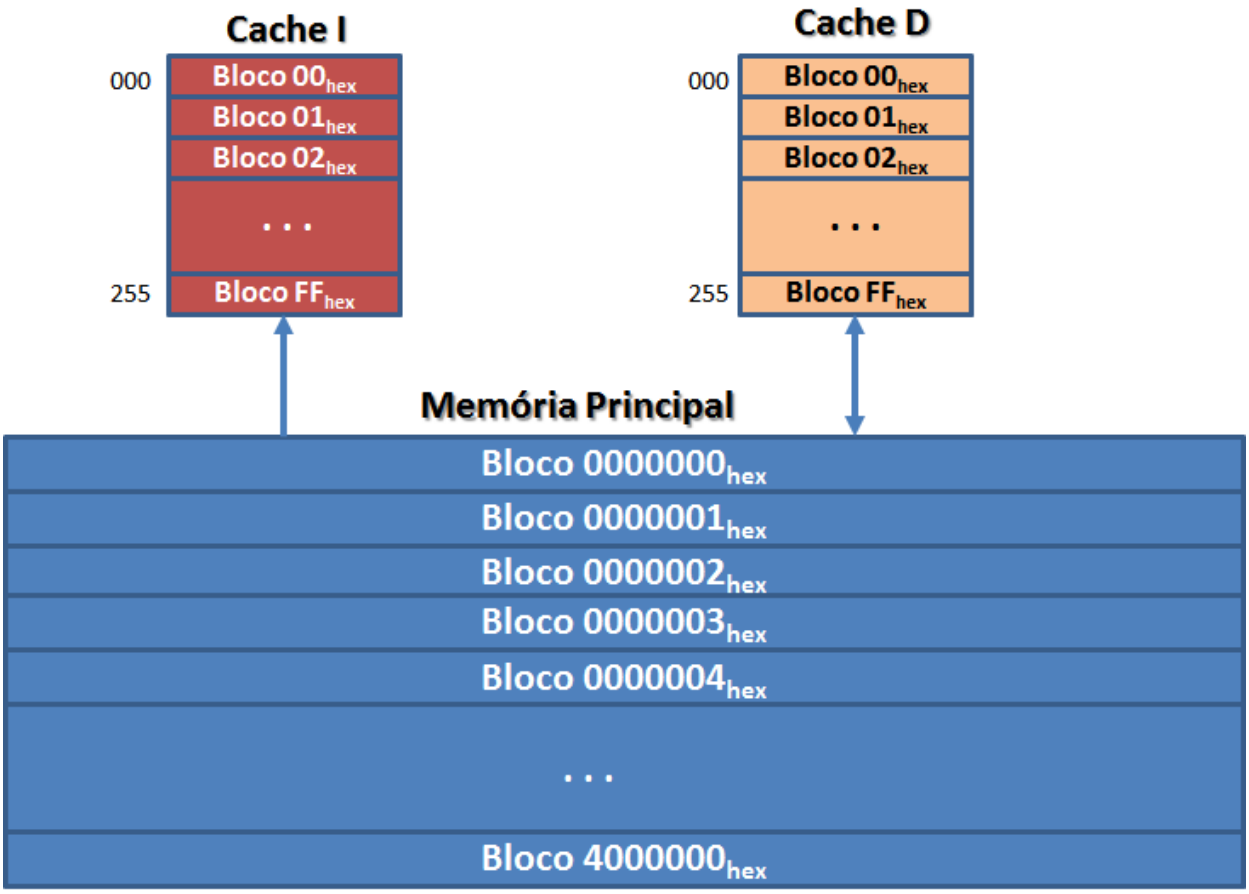
2.2 – Hierarquia de Memória

A hierarquia de memória será descrita de maneira paramétrica, uma vez que cada grupo de trabalho receberá parâmetros diferentes.

- *Memória Principal (MP)*
  - *Possui somente um módulo de memória contendo todos os blocos endereçáveis.*
  - *A MP possui 2^{30} palavras (no caso deste projeto a MP deverá possuir 2^{14} palavras) de 32 bits (4 bytes).*
  - *Latências:*
    - *100 ns por ciclo de acesso (leitura ou escrita).*
- *Cache I*
  - *Possui 16 KBytes ou 4096 palavras de 32 bits*
  - *16 palavras por bloco*
  - *Mapeamento direto*
  - *Tempo de acesso = 5 ns (para leitura ou escrita)*

- *Cache D*
  - *16KBytes ou 4096 palavras de 32 bits*
  - *16 palavras por bloco*
  - *Algoritmo de mapeamento de bloco: Associativo por conjunto de 2 blocos*
  - *Algoritmo de substituição de bloco: LRU*
  - *Algoritmo de escrita de bloco: Write Back com um buffer de escrita de uma palavra*
  - *Tempo de acesso = 5 ns (para leitura ou escrita)*

O esquemático abaixo representa a hierarquia de memória na MA especificada aqui.



2.3 – Interrupções

O tratamento de interrupções é uma função primordial do processador. O aluno deverá recorrer a descrição deste projeto no livro texto para definir e implementar o procedimento de interrupção.

2.4 – Estado inicial

- *Sempre que for "ligado" ou logo após um "reset" (que deve ser um sinal assíncrono externo, também modelado na descrição comportamental em VHDL), o processador deve retornar a um estado conhecido, especificado abaixo.*
  - *O pc (contador de instruções) deve conter o valor 0x0000*
  - *Os registradores 0 a 3 do BR (Banco de registradores) devem conter o valor 0x0000*
  - *Registrador 4 ($gp) deve conter o valor 0x10008000*
  - *Registrador 5 ($sp) deve conter o valor 0xffff*
  - *Registrador 6 ($fp) deve conter o valor 0x7ffff*
  - *Registrador 7 ($ra) deve conter o valor 0x0000*
  - *Todos os caches devem ter os seus conteúdos igualados a zero.*
  - *A memória principal terá os valores de cada um dos seus endereços retornados a um estado inicial bem conhecido correspondente ao programa de teste (SORT) já escrito pelos alunos numa tarefa anterior.*

2.5 – Detalhes para codificação VHDL

A MP deverá ser implementada como um arquivo texto (com a extensão txt) de múltiplas linhas, sendo cada linha apresentada no seguinte formato:

- *<endereço>|<valor>|<comentário>*

Os três campos de cada linha tem o seguinte formato.

- *O campo <endereço> deverá conter uma cadeia de caracteres ASCII que representa um endereço binário de 32 bits (i.e. uma cadeia de „0"s e „1"s).*
- *O campo <valor> também deverá conter uma cadeia de caracteres ASCII que representa um valor binário de 32 bits (i.e. uma cadeia de „0"s e „1"s).*
- *O campo <comentário> pode conter uma cadeia de caracteres ASCII alfanuméricos.*

Este arquivo texto é esparso: apenas os endereços cujos valores são diferentes de zero devem estar representados por linhas individuais explicitando tais valores. Um conjunto de linhas que explicita o conteúdo da MP no início da operação do processador (i.e. logo após este ser

“ligado) deve fazer parte do arquivo texto em seu “estado inicial”, o mesmo ao qual o processador deve retornar apos um “reset”.
Finalmente, é importante ressaltar que não há necessidade de qualquer ordenação no arquivo texto que implementa esta MP, dado que a localização da mesma se dará diretamente, sendo esta uma consulta a uma tabela do tipo <chave>,<valor> (onde <chave> é o <endereço> e <valor> é o <conteúdo>)
Um exemplo deste arquivo texto é o que segue (perceba que a cadeia LXYZ NÃO FAZ PARTE DO ARQUIVO!):

```
...
L010:01000000000000000001000000000000|00000000101000010000000000011000|muli   $2,$5,4
L011:01000000000000000001000000000100|00000000000110000001100000100001|add    $2,$4,$2
L012:01000000000000000001000000001000|10001100011010000000000000000001|lw $15, 0($2)
...
```

3 – Caracterização

3.1 – Métricas

Para a caracterização de desempenho do processador, deve-se coletar dados relevantes durante a execução de programas específicos com este propósito (o de caracterização). A coleta de dados deverá ser executada através da leitura de “sondas de desempenho”, que assumem a forma de registradores de propósito especial que deverão fazer parte da descrição comportamental do processador codificada em VHDL. As “sondas de desempenho” devem ser as que seguem.

- *Número de ciclos transcorridos.*
- *Número de instruções executadas.*
- *Número de bolhas no pipeline.*
- *Cache “hit ratio” para os caches L1I e L1D.*
- *Número de acessos à memória (Cache ou Principal).*
- *Tempo de execução do programa SORT.*
- *Frequência de ocorrência das instruções na execução do programa SORT.*

- *Número médio de ciclos por instrução do programa SORT.*

3.2 – Cargas de Trabalho

3.2.1 – Programas selecionados

Oprograma utilizado para o teste do processador FemtoMips é o programa SORT já desenvolvido pelos alunos numa tarefa anterior.

3.2.2 – Conjuntos de dados

Os conjuntos de dados serão compostos por estruturas de dados (vetores e matrizes) cujos valores serão gerados aleatoriamente. Cada conjunto de dados será gerado com tamanhos variáveis.

3.2.3 – Sessão de Caracterização do Processador no relatório final

Além do relatório de documentação do projeto que vem sendo construído ao longo da execução do projeto, os alunos devem incluir uma sessão de caracterização do processador projetado.
A caracterização será realizada pela execução da carga de trabalho, tendo estas como fator o tamanho do conjunto de dados e como níveis cada um dos tamanhos selecionados. O relatório de caracterização deverá então conter as seguintes informações.

- *Caracterização de desempenho do processador, com ao menos os seguintes gráficos e/ou tabelas.*
  - *CPI x Carga de Trabalho X Tamanho do Conjunto de Dados*
  - *"Hit Ratio x Carga de Trabalho X Tamanho do Conjunto de Dados*
  - *"Acesso a Memória" x Carga de Trabalho X Tamanho do Conjunto de Dados*
- *Uma análise crítica do comportamento do processador na execução de várias cargas de trabalho e vários tamanhos dos conjuntos de dados.*

Todos os dados devem ser adequadamente tratados do ponto de vista estatístico, incluindo o número de execuções, a média e o coeficiente de variação. O relatório e o código VHDL devem ser submetidos pelo site do TIDIA-Ae na ferramenta "Atividades".