0.1 Microprocessadores

0.1.1 Definição

Segundo [7], um microprocessador é um circuito eletrônico muito complexo composto de milhares de transistores microscópicos num único circuito integrado contendo até cerca de 40 terminais. Os milhares de transistores que compõem o microprocessador são arranjados para formar muitos circuitos diferentes dentro do chip. Entre estes circuitos pode-se destacar registradores, decodificadores, contadores, etc.

O coração de um microcomputador é sua unidade de processamento (MPU). A MPU de um microcomputador é implementado com um dispositivo VLSI (*Very Large Scale Integration*) conhecido como microprocessador, ou somente processador, sendo mais direto. Um microprocessador é uma unidade de processamento de propósito geral construído em um único circuito integrado (CI), [6].

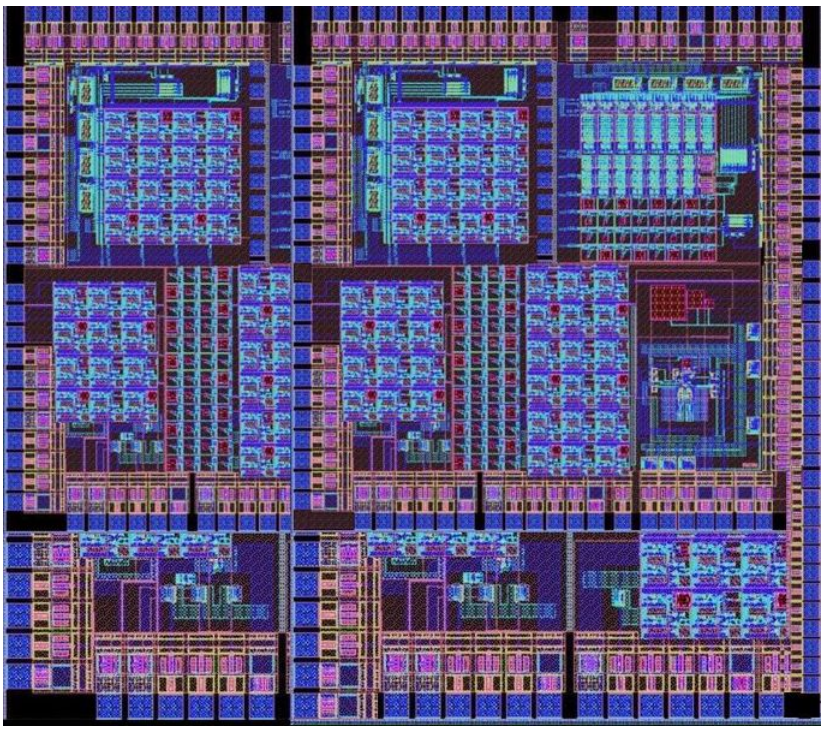

Figura 1 – Microcircuito produzido pelo processo fotográfico de multicamadas. Microprocessador com frequência de 4.8GHz, utilizado para o processamento de imagens do Gyroscan 6,2 Tesla. [1]

Como visto acima sabemos que o microprocessador é o coração de um sistema microprocessado, na figura 2, defini-se as quatro partes básicas de um sistema microprocessado, que incluem um microprocessador, memória e entrada/saída que são interligados por um sistema de *buses*, que será explicado mais a frente. Um *bus* é um conjunto de fios que transmitem informação entre dois ou mais dispositivos[9].

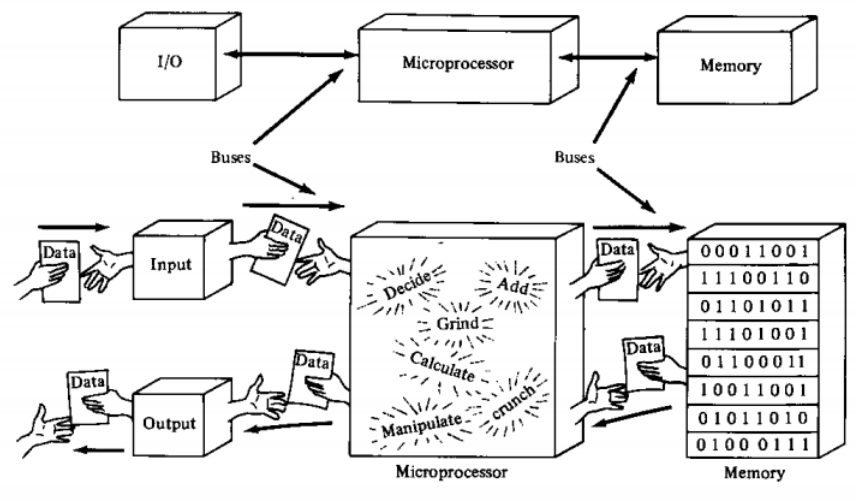

Figura 2 – Diagrama de blocos de um sistema microprocessado. [9]

0.1.2 Funcionamento

Os microprocessadores funcionam à partir de um relógio interno, feito de quartz que quando sujeito a uma corrente elétrica, emite pulsos, chamados de "top". Tais pulsos fazem com que o microprocessador execute uma ação, ou seja, uma instrução, seja a mesma executada de forma parcial ou total. Estes pulsos também define a potência do microprocessador, sendo esta potência é definida como o número de instruções executadas pelo por segundo e tem como unidade utilizada o MIPS (Milhões de Instruções Por Segundo) [8].

Existem dispositivos de entrada e saída que permite a importação de dados para armazenamento ou processamento, a exportação dos resultados e acessos aos dados armazenados. Outros sinais importantes para o funcionamento do microprocessador é o sinal de *reset*, que faz com que a CPU volte a um estado inicial que é definido e conhecido, voltando a este estado o microprocessador pode começar a executar programas, e o sinal de interrupção que faz com que o microprocessador pare sua execução e comece a executar uma rotina pré-definida [8]

0.1.3 Programa de Computador

A figura 3 é uma representação visual de um programa de computador, onde ter-se o código não é o suficiente. Para realizar a tarefa especificada pelo programa, o computador (microprocessador) necessita ler as instruções do programa, interpretá-las e executá-las. [9].

De acordo a [9] a maneira que um computador executa um programa é cíclica e segue a seguinte ordem:

1. Leitura (*Fetch*) de uma instrução

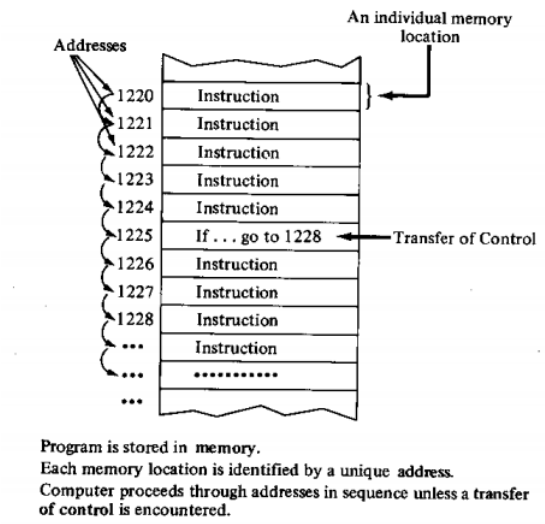

Figura 3 – Organização de um programa de computador [9]

Onde o computador lê uma instrução e copia a instrução da memória para o seu cérebro (*Microprocessador*).

2. Interpretação (*Decode*) da instrução
   Cada número que representa uma instrução dentro do programa, possui um certo significado para o computador, em termos de qual ação que deve ser realizada.

3. Execução (*Execute*) da instrução

Para a realização deste ciclo, o microprocessador conta com uma série de circuitos internos com funcionalidades específicas, que serão descritos à seguir.

0.1.4 Registradores

Os registradores, são utilizados para salvar informação binária durante o tempo de execução de um programa. Cada registro possui uma função específica associada a ele [10]:

O **acumulador**, é um registro primário associado a uma *ALU* (Unidade Lógica Aritmética) e as operações de entrada/saída. O **registro de instrução**, guarda o código binário da instrução que está sendo executada. O **contador de programa**, contém o endereço de memória da próxima instrução que deve ser tomada.

Todos estes registradores podem ser visualizados na figura 4.

0.1.5 Unidade Lógica Aritmética

A Unidade Lógica Aritmética, ou *Arithmetic Logic Unit* (ALU) em inglês, é o maior componente da unidade central de processamento de um sistema microprocessado. A

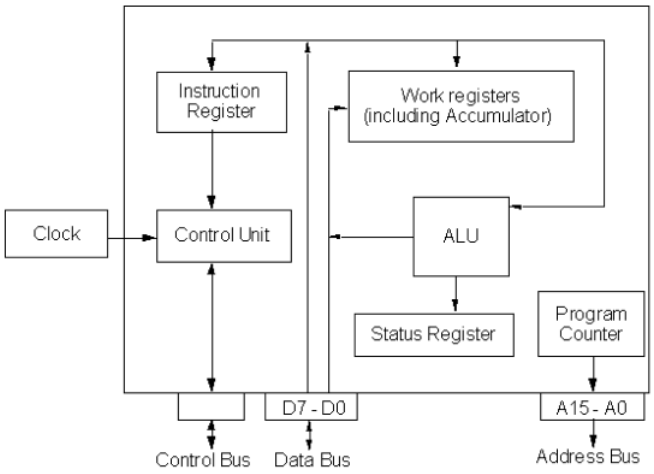

Figura 4 – Organização interna de um microprocessador hipotético [10]

unidade realiza todos os processos relacionados a operações aritméticas e lógicas que necessitam ser feitas nas instruções. Em alguns microprocessadores a ALU é divida em unidade aritmética (UA) e unidade lógica (UL). Uma ULA pode ser desenvolvida por engenheiros para calcular qualquer operação. Assim que as operações começam a ficar mais complexas, a ULA fica mais cara, ocupa mais espaço e dissipa mais calor. Por isto engenheiros fazem a ULA poderosa o suficiente, para garantir que a Unidade de Processamento seja também poderosa e rápida, porém não tão complexa, que a torne proibitiva em termos de custo entre outras desvantagens [13].

ULAs normalmente realizam as seguintes operações:

- **Operações Lógicas**: Essas incluem AND, OR, NOT, XOR, NOR, NAND, etc.
- **Operações de Rotacionamento de Bits**: Pertence ao rotacionamento da posição dos bits por um certo número de rotações para direita ou esquerda, que são consideradas operações de multiplicação.
- **Operações Aritméticas**: Refere-se a adição e subtração de bit. Entretanto multiplicação e divisão são as vezes implementadas, essas operações são mais caras de se realizar. A adição pode ser utilizada como substituta para a multiplicação e a subtração para a divisão.

0.1.6 Unidade de Controle

A unidade de controle é composta por um controlador de sequência e um decodificador de instrução. Durante a execução, a unidade de controle, ajusta o conteúdo do contador de programa para ser posicionado nas linhas de endereçamento. Essas linhas indicam o endereço da posição de memória, que contém o código da próxima instrução a ser

executada. Em seguida, a unidade de controle insere no registrador de instrução o código de instrução da posição de memória. O decodificador de instrução é habilitado e a unidade de controle ativa as linhas de controle necessárias, buscando dos resultados desejados [10].

0.1.6.1 Sinais de Controle

Os sinais de controle são sinais elétricos que orquestram as diversas unidades do processador, que participam na execução de uma instrução. Os sinais de controle são distribuídos devido a um elemento chamado sequenciador. O sinal *Read/Write*, em português Leitura/Escrita, diz para a memória ou outros dispositivos que o processador quer ler ou escrever uma informação [8].

0.1.7 Sistema de Barramentos

No diagrama simplificado da figura 5 todos os módulos lógicos se comunicam com a Unidade Central de Processamento. Na prática, muitos modelos de interconexão podem ser usados, geralmente através de barramentos. Lembre-se de que um barramento é um meio de transmissão de informações ou sinais, distinguidos por suas funções. No caso dos sistemas baseados em microprocessador, ao menos três barramentos são fornecidos [5]:

- **Barramento de Dados**: Transmite dados entre as unidades. Portanto, um microprocessador de 8 bits requer um barramento de dados de 8 linhas para transmitir dados de 8 bits em paralelo. Semelhantemente, um microprocessador de 64 bits necessita de um barramento de dados de 64 linhas para transmitir dados de 64 bits em paralelo. Se o barramento de dados para um microprocessador de 64 bits fosse formado por 8 linhas, seriam necessárias oito transmissões sucessivas, tornando mais lento o sistema. O Barramento de Dados é bi-direcional, isto é, pode transmitir em ambas as direções.

- **Barramento de Endereço**: É usado para selecionar a origem ou destino de sinais transmitidos em um dos outros barramentos ou numa de suas linhas, conduzindo endereços. Uma função típica do Barramento de Endereço é selecionar um registrador em um dos dispositivos do sistema, que é usado como a fonte ou o destino do dado. O Barramento de Endereço do nosso computador padrão, tem 16 linhas e pode endereçar 216 (64 K) dispositivos (1K = 1024, ou 210 , no jargão de computação).

- **Barramento de Controle**: Sincroniza as atividades do sistema, conduzindo o status e a informação de controle de/para o Microprocessador. Para um Barramento de Controle ser formado, ao menos 10 (geralmente são mais) linhas de controle são necessárias.

De acordo a [5], os barramentos são implementados como linhas de comunicação reais. Eles podem ser posicionados como parte do circuito no próprio Chip (Barramentos internos) ou podem servir de comunicação externa entre os Chips (Barramentos externos). Os barramentos externos podem ser expandidos para facilitar a conexão de dispositivos especiais. Um projeto eficiente de barramentos é crucial para a velocidade do sistema.

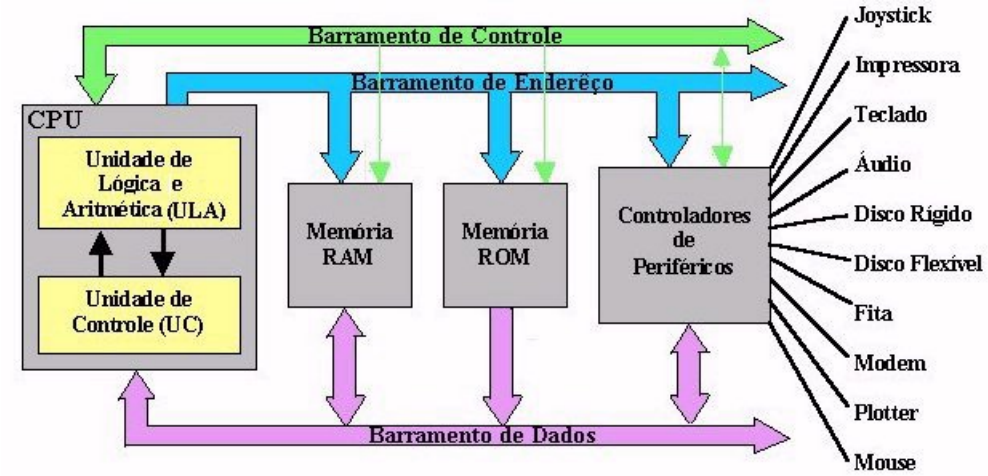

Figura 5 – Diagrama simplificado de um Microcomputador [5]

0.1.8 Dispositivos de Entrada/Saída

Os dispositivos de entrada/saída (E/S) ou *input/output (I/O)*, são também denominados periféricos, eles permitem a interação do processador com o homem, possibilitando a entrada e/ou saída de dados. Porém, é necessário o módulo mais a direita da figura 5, que são os controladores de periféricos. Tais controladores possuem a tarefa de combinar as velocidades entre os dispositivos, pois a maioria dos periféricos são consideravelmente mais lentos que a unidade de processamento, convertem dados de um formato em outro [5]. Exemplos de periféricos de entrada: teclado, mouse, scanner, etc. Dispositivos de saída: monitor, impressora, etc.

0.1.9 Arquiteturas

De acordo com [4], uma das mais importantes abstrações é a interface entre o hardware e o software de baixo nível. Por causa de sua importância, é dado uma nomenclatura especial: **arquitetura do conjunto de instruções** (ISA), ou simplesmente arquitetura de uma máquina. O conjunto de instruções, inclui qualquer coisa que programadores necessitam para saber como programar em linguagem de máquina corretamente, incluem instruções, dispositivos E/S, entre outros. Tipicamente o sistema operacional irá encapsular os detalhes da realização da E/S, alocação de memória, e outras funcionalidades

de baixo nível do sistema, portanto, programadores não precisam se preocupar com estes detalhes. Dois tipos de conjuntos de instruções existentes serão explicados a diante.

0.1.9.1 CISC - Complex Instruction Set Computer

CISC é uma arquitetura de processador, que teve como princípio o uso eficiente de memória e a facilidade de programar. Cada instrução desse processador tem várias operações em seu interior ajudando o programador a implementar programas. A maioria dos projetos de microprocessadores comuns - incluindo o Intel (R) 80x86 e séries Motorola 68K - também seguem a filosofia CISC [3].

Os primeiros processadores utilizados para decodificar e executar cada instrução no conjunto de instrução do processador, principalmente para trabalhos simples, com poucos registros, funcionou, mas não para sistemas complexos. Assim, os seus criadores construíram uma lógica simples de controlar os caminhos de dados entre os vários elementos do processador, e usou um conjunto simplificado de instruções de microcódigo para controlar a lógica do caminho de dados.

A microprogramação é uma representação simbólica do controle em forma de instruções, chamadas microinstruções, que são executadas em uma micromáquina simples [4], podemos ver na figura 6 o exemplo de um microcódigo.

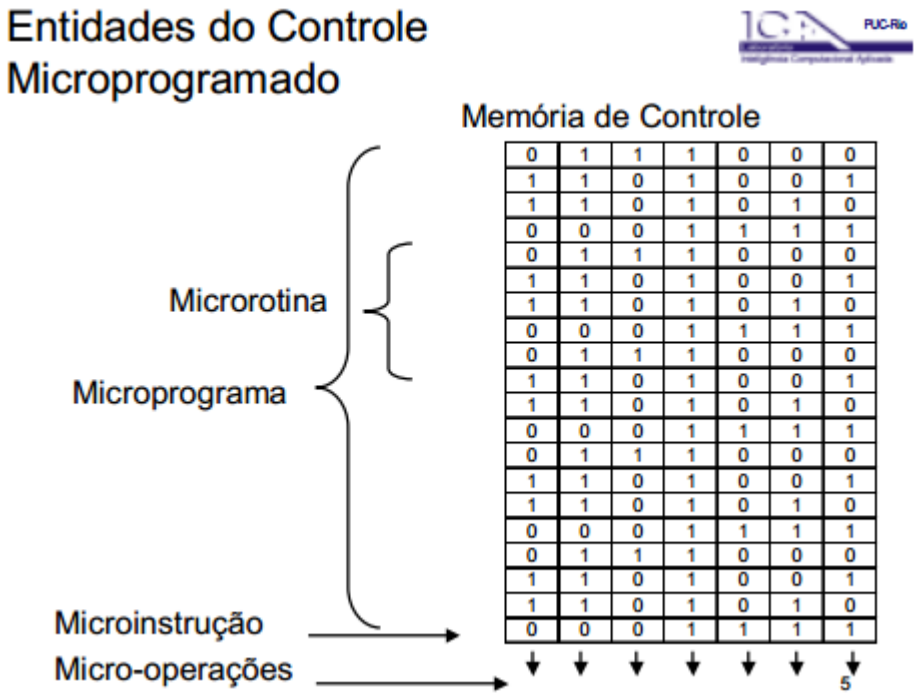

Figura 6 – Entidades do Controle Microprogramado [11]

0.1.9.2 RISC - Reduced Instruction Set Computer

Na década de 1980, ocorreu as mudanças para a nova arquitetura, o modelo RISC (Reduced Instruction Set Computer). As melhorias nas linguagens de programação, tecnologia de compiladores e custo de memória significaram que, menos programação estava sendo feita no nível do assembly, de modo que os conjuntos de instruções poderiam ser medidos pela forma como os compiladores os usavam, ao contrário de como os programadores em assembly os usavam. Praticamente todos os novos conjuntos de instruções desde 1982, seguiram essa filosofia RISC de tamanhos de instrução fixos, conjuntos de instrução load/store, modos de endereçamento limitados e operações limitadas. ARM, Hitachi SH, IBM PowerPC, MIPS e Sun SPARC são todos exemplos de arquiteturas RISCs [4].

0.1.9.3 Comparação entre RISC e CISC

Como visto anteriormente, a arquitetura CISC apresenta instruções complexas executadas em vários ciclos de clock, enquanto a arquitetura RISC possui somente instruções que são executadas em apenas um ciclo. No quesito de acesso a memória, o conjunto complexo possui vários tipos de modos de endereçamento de memória, facilitando o trabalho do programador. Os microprocessadores RISC são considerados máquinas *load/store*, o que é possível pois ele possui uma grande quantidade de registradores dos mais variados tipos. A clara vantagem da arquitetura RISC é em questão de velocidade, pois por possuir um conjunto de instruções, com todas instruções com formato fixo, ocorre um uso intenso de *pipeline*. No desenvolvimento de um microprocessador CISC, a complexidade do sistema se encontra no microprograma, como visto um exemplo na figura 6, e na arquitetura RISC a complexidade se encontra no compilador. Ambas arquiteturas são muito bem aceitas no mercado e cada uma possui suas vantagens e desvantagens para serem aplicados em diversos tipos de projetos.

0.1.10 Memória Cache

De acordo com [12], a memória cache consegue realizar a ponte entre a diferença de velocidade entre o processador e a memória. A cache é um pequeno espaço de alta velocidade que se situa entre o processador e a memória na hierarquia de memórias.

Cache, foi o nome escolhido para representar o nível na hierarquia de memória entre o processador e a memória do primeiro computador comercial a ter este nível extra, como pode ser visto na Figura 7 item (b). A razão da cache ($SRAM$) ser menor é devido a maiores decodificadores de endereço, pois são mais lentos do que menores decodificadores de endereço. Quanto maior a memória é, mais complexo é seu decodificador de endereço, e mais tempo leva para identificar o valor da posição de memória do endereço desejado [12].

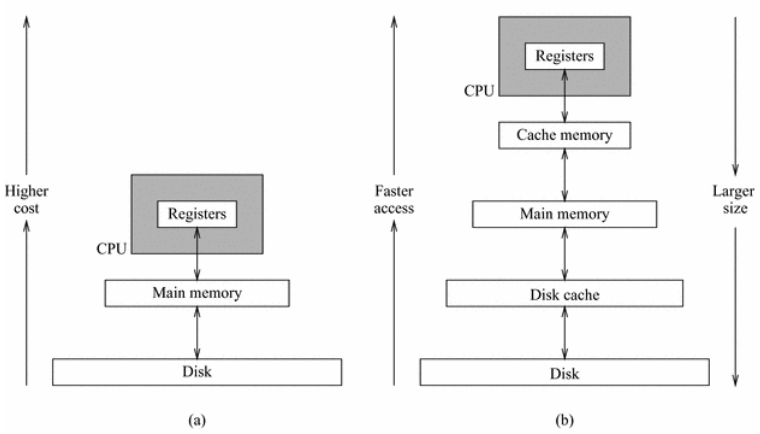

Figura 7 – Hierarquia de Memórias [12]

É possível utilizar este conceito e dar um passo a diante introduzindo uma *SRAM* menor entre o cache o e processador, dentro do próprio envólucro do processador, criando dois níveis de memória cache L1 e L2, como podemos ver na figura 8 [12].

Figura 8 – Dois níveis de memória cache L1 e L2 [12]

0.1.11 Pipeline

0.1.11.1 Definição

De acordo com [4],*Pipelining* é uma técnica de implementação no qual multiplas instruções são sobrepostas durante a execução, como visto na figura 9. Hoje em dia, *pipelining* é a chave para fazer processadores rápidos [4].

Como podemos ver, a utilização de *pipeline* torna a execução muito mais rápida do que se as tarefas fossem executadas sequencialmente. Como exemplo utilizaremos o microprocessador MIPS, que possui 5 estágios de execução de uma instrução, sendo elas: Decodificação da Instrução (*Instruction Fetch)*, Leitura dos Registros (*Reg*),Operação de ULA (ALU) , Acesso ao dado e Escrita no Registro. Na figura 10 podemos ver quantitativamente a diferença do processo com utilização de *pipeline*.

0.1.11.2 Desenvolvimento de um conjunto de instrução para o *Pipeline*

Primeiramente, todas as instruções do MIPS possuem o mesmo comprimento, esta restrição faz com que fique mais fácil decodificar as instruções no primeiro estágio e n segundo

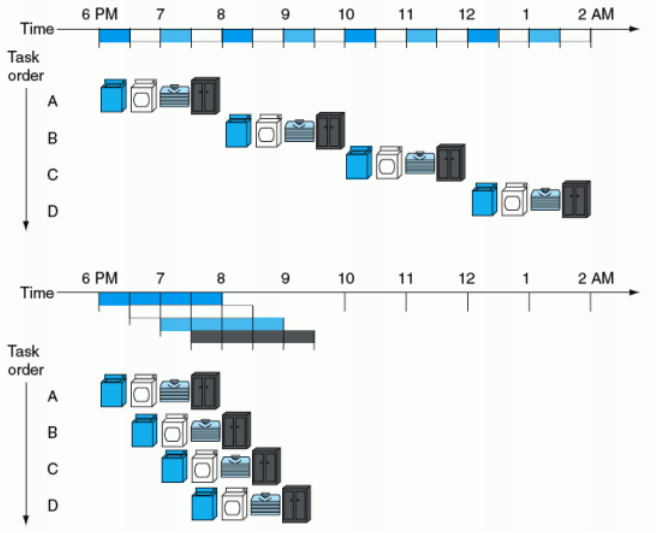

Figura 9 – A analogia de uma lavanderia com o pipeline [4]

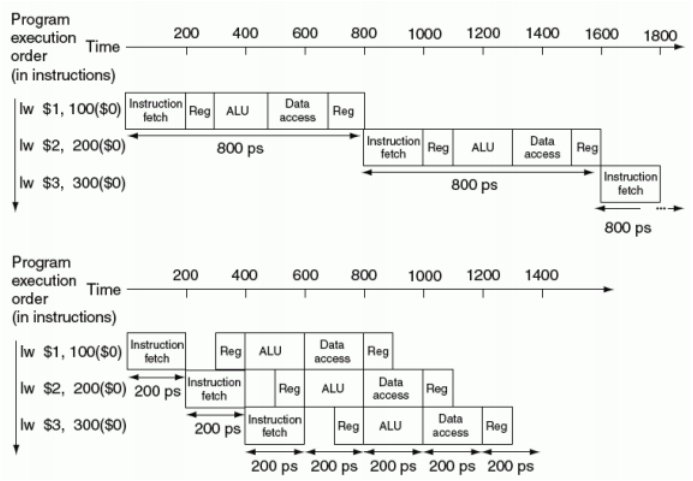

Figura 10 – Comparação quantitativa da utilização de *Pipeline* utilizando a instrução *Load Word* do microprocessador MIPS. [4]

estágio do *pipeline*. Em um conjunto de instruções, como o do IA-32, onde instruções variam entre 1 byte até 17 bytes, o *pipelining* é consideravelmente mais complicado [4]. Atualmente a arquitetura do IA-32 transforma as instruções em microinstruções, sendo essas realmente que serão utilizadas para a realização do *pipeline* com uma arquitetura *CISC*.

0.1.11.3 Problemas do*Pipeline*

O **primeiro** tipo de problema do *pipeline*, é o problema estrutural, que significa que o hardware não suporta a combinação de instruções ao qual desejamos que sejam executadas em um único ciclo de clock. Ao caso de termos somente uma única memória, no segundo e quarto passo, são necessários acessos a memória que não podem ser executadas de uma única vez [4]. O **segundo** tipo de problema , quanto a acesso aos dados, ocorre quando

o *pipeline* fica travado no momento em que ocorre uma etapa deve esperar outra etapa completar para continuar, por exemplo no seguinte trecho de código do MIPS, na figura 11.

```
add    $s0, $t0, $t1
sub    $t2, $s0, $t3
```

Figura 11 – Trecho de código que exemplifica um problema do *pipeline* [4].

Quando este tipo de problema ocorre, um fato chamado *bubble* aparece no meio do *pipeline*, como se fosse uma linha vazia entre dois processos do *pipeline*, podendo ser notado na figura 12.

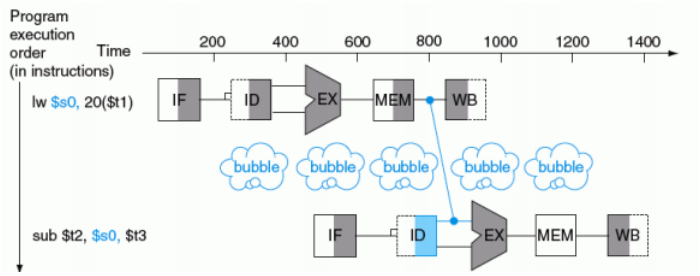

Figura 12 – Fenômeno do tipo *bubble* no *pipeline* semelhante ao problema da figura 11 [4].

Para resolver este tipo de problema, o código pode ser reescrito de uma forma que evite a dependência das informações, essa correção pode ser realizada tanto pelo compilador como pelo programador.

O terceiro tipo de problema, é chamado de problema de controle, também chamado de problema de *branch*, surge da necessidade de tomar uma decisão baseada nos resultados de uma instrução enquanto outras estão em execução [4].

As instruções de *branch* são instruções de desvios condicionais no código, portanto a execução da próxima instrução depende se o *branch* causará desvio ou não, caso o desvio ocorra, o fenômeno de *bubble* ocorre novamente, pois o *pipeline* necessita ficar um tempo parado esperando a tomada de decisão, semelhante a figura 13.

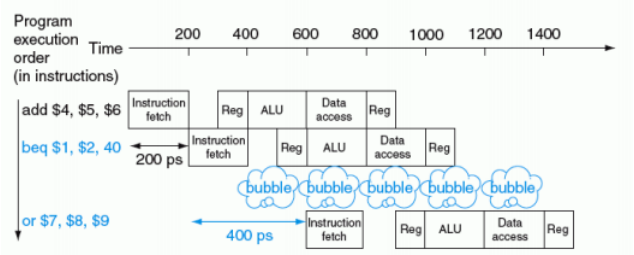

Figura 13 – Fenômeno do tipo *bubble* no *pipeline* quando ocorre o problema de *branch* [4].

Para solucionar este problema, desenvolveu-se uma estrutura dentro do próprio microprocessador, chamada *branch predictor*. O *branch predictor*, é um circuito digital que tenta adivinhar qual vai ser o caminho que o branch irá seguir, para continuar preenchendo o *pipeline*. Hoje em dia, tal estrutura desempenha um papel fundamental para o desenvolvimento de processadores de alta performance.

0.1.12 Processadores Multi-Core e Hyper-Threading

De acordo com [2], basicamente, multi-core é um design ao qual um único processador físico, contém o núcleo lógico de mais de um processador, como pode ser visto na figura 14. O objetivo deste design é habilitar o sistema a executar mais tarefas simultaneamente e desse modo alcançar uma maior performance do sistema.

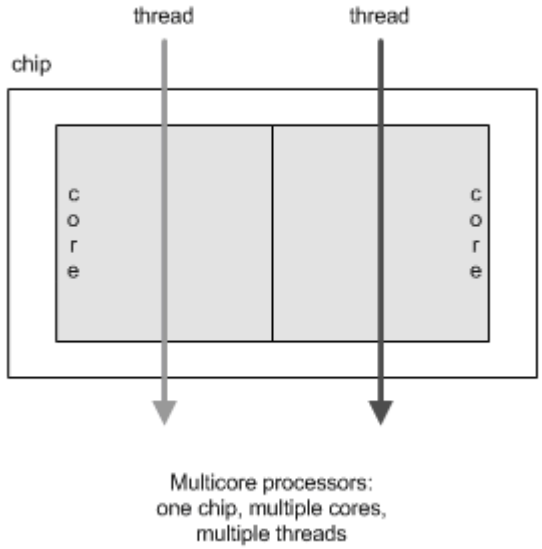

Figura 14 – Processador com dois núcleos dentro de um único processador [2].

Programas são executados, à partir de threads, essas threads são sequências de instruções relacionadas. Nos primórdios do PC, maioria dos programas consistia de uma única thread, o sistema operacional naquela época era capaz de executar somente um programa por vez, tendo como resultado uma sensação dolorosa que seu PC congelava enquanto imprimia um documento ou uma folha de trabalho, o sistema era incapaz de realizar duas tarefas simultaneamente. Inovações no sistema operacional introduziram os sistemas multitarefa, no qual um programa pode ser brevemente suspendido enquanto executa outra, de uma maneira que o usuário não perceba, pealizando esta troca rapidamente, o sistema tem a aparência de estar executando os programas simultaneamente, contudo o processador estava de fato, o tempo executando uma única thread.

No início dos anos 2000, o design de processadores ganhou recursos adicionais, como uma lógica dedicada para operações com ponto flutuante, para suportar a execução de

múltiplas instruções em paralelo. A Intel®, definiu que o melhor uso desses recursos empregando-as para executar duas threads simultaneamente no mesmo núcleo de processamento, figura 15. A Intel®nomeou este procedimento simultâneo como *Hyper-Threading Technology*®e lançou-a nos processadores **Intel Xeon ®** em 2003. De acordo com medidores da Intel®, aplicações que eram escritas utilizando múltiplas threads realizaram suas tarefas 30% mais rápido do que se for executado utilizando a tecnologia **HT**. Para induzir o sistema operacional a reconhecer um processador como duas possibilidades de execução de *pipeline*, *chips* foram feitos para aparentar ser dois processadores lógicos.

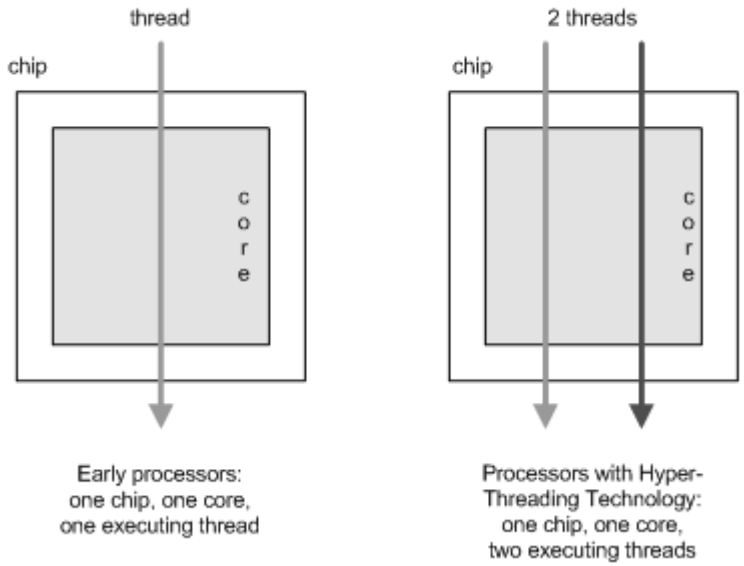

Figura 15 – Exemplo de um processador com a tecnologia *Hyper-Threading* [2].

Referências

[1] Angeloleithold. Internal integrated circuit. http://pt.wikipedia.org/wiki/Ficheiro:InternalIntegra Dezembro 2004.

[2] A. Binstock. Multi-core processor architecture explained. http://software.intel.com/en-us/articles/multi-core-processor-architecture-explained, Setembro 2013.

[3] CISC. Cisc. http://www.laynetworks.com/CISC.htm, Agosto 2013.

[4] J. L. H. David A. Patterson. *Computer Organization and Design, The Hardware/Software Interface*. Elsevier, Oxford, Reino Unido, 2005.

[5] P. M. R. de Gouveia Nóbrega Filho. Fundamentos de hardware. http://www.di.ufpb.br/raimundo/ArqDI/Arq5.htm, Agosto 2013.

[6] W. A. T. e Avtar Singh. *The 8088 and 8086 Microprocessors, Programming, Interfacing, Software, Hardware, and Applications*. Pearson, Upper Saddle River, New Jersey, Columbus, Ohio, 1947.

[7] F. B. M. e Ignácio Sérgio M. Ferreira. *Práticas com Microprocessadores*, volume I.

[8] Microprocessadores. Microprocessadores. http://pt.kioskea.net/contents/pc/processeur.php3, Agosto 2013.

[9] S. B. Newell. *Introduction to Microcomputing*. John Wiley and Sons, Inc., 1989.

[10] I. D. of Industrial Engineering and L. Management. Lecture 7: Microprocessor structure, assembly programming. http://www.ielm.ust.hk/dfaculty/ajay/courses/alp/ieem110/lecs/mup/mup.html, Agosto 2013.

[11] PUC-Rio. Puc-rio. http://tcs.eng.br/PUC/plog/sd11-Microprogramacao.pdf, Agosto 2013.

[12] D. Tarnoff. *Computer Organization and Design Fundamentals*. Lulu.com, 2007.

[13] Techopedia. Arithmetic logic unit (alu). http://www.techopedia.com/definition/2849/arithmetic logic-unit-alu, Agosto 2013.