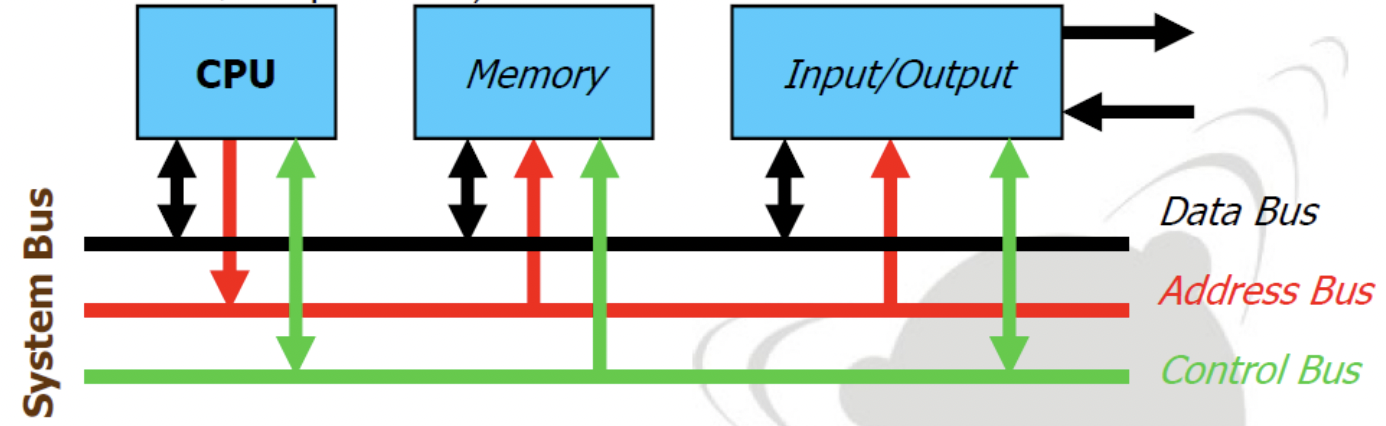
Modelo de Von Newman



Data Bus: barramento de transferência de informação (CPU << Memória, CPU << Input/Output).

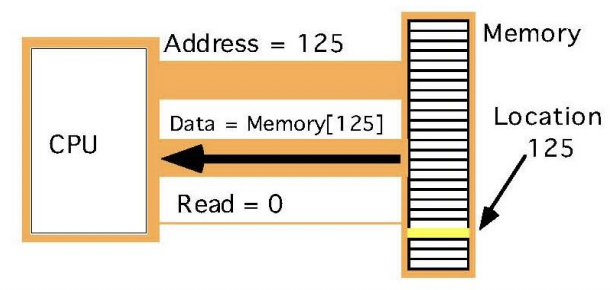
Address Bus: identifica a origem/destino da informação.

Control Bus: sinais de protocolo que especificam o modo como a transferência de informação deve ser feita.

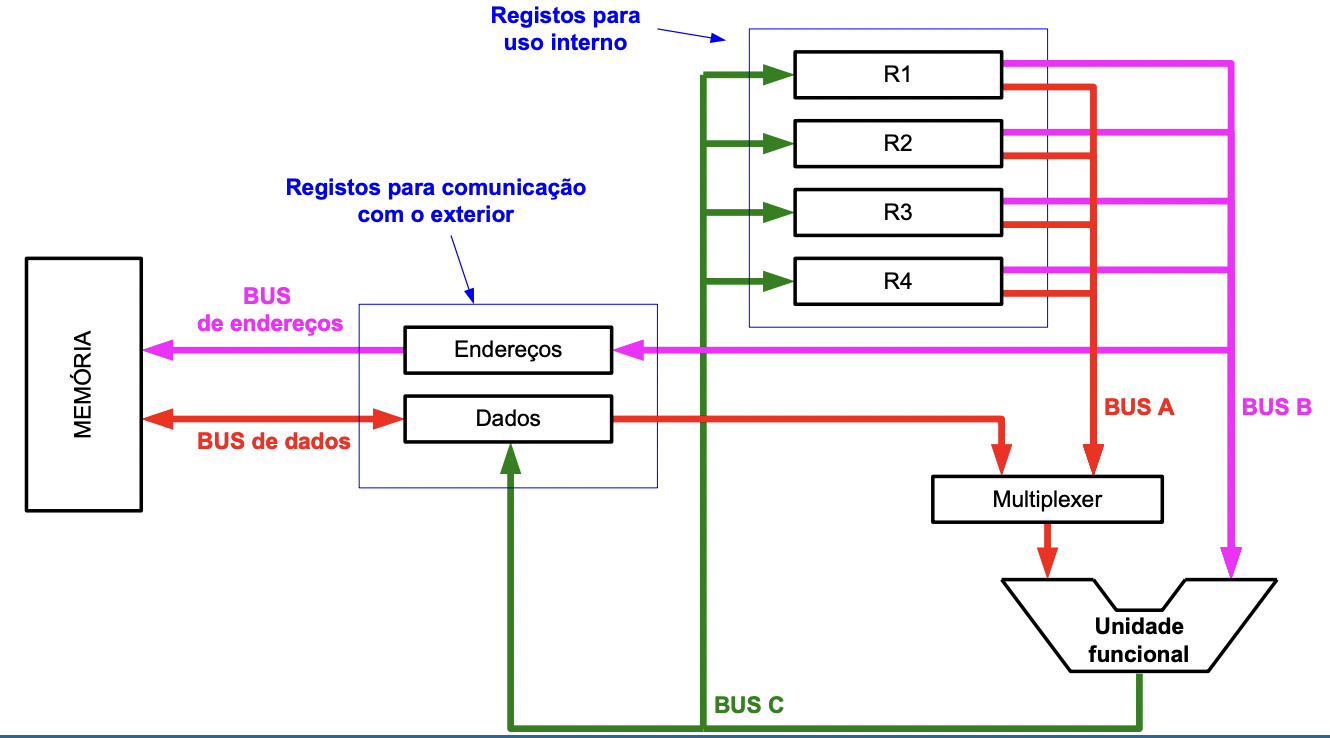
Operação de Escrita



Operação de Leitura



Estrutura Básica de um CPU



* Cada um dos R’s pode ser redirecionado
* Funcionam como multiplexers:
  + Seleciona-se um valor e reencaminha-se para o output

CPU

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Registos** | **Bus Dados** | **Bus Endereçamentos** | **MIPS** | **Transístors** |
| **4004** | 4 bits | 4 bits | 12 bits (4k) \* | 0,06 | 2300 |
| **8008** | 8 bits | 8 bits | 14 bits (16k) | 0,06 | 3500 |
| **8080** | 8 bits | 8 bits | 16 bits (64k) | 0,64 | 4500 |
| **8086** | 16 bits | 16 bits | 20 bits (1M) | 0,75 | 29000 |
| **8088** | 16 bits | 8 bits | 20 bits (1M) | 0,75 | 29000 |

1024 = 1k

|  |
| --- |
| CÓDIGO |
|  |
|  |
|  |
|  |
|  |
| DADOS |
|  |
|  |
|  |
|  |
|  |
| PILHA |
|  |
|  |
|  |
|  |
|  |
|  |
| EXTRA |
|  |
|  |
|  |
|  |

8 bits 8 bits

|  |  |
| --- | --- |
| AH | AL |
| BH | BL |
| CH | CL |
| DH | DL |
| BP | |
| SP | |
| SI | |
| DI | |

64KB

18297 ⬄ CS> 15CCFh:25A7h

20 bits vs 16 bits

É necessário fazer uma conversão

IP = 25A7h

CS = 15CFh

CS = 15CFh

x 10

\_\_\_\_\_\_\_\_\_

15CF0h

+ 25A7h

\_\_\_\_\_\_\_\_\_

18297h

Endereço Físico

15CFh : 25A7h > Endereço Lógico

8088

Usado como processador “low cost”.

O processador em si tem o mesmo preço, porém os pc’s eram mais baratos.

80286

Já houve um grande avanço tecnológico, porém a memória (16MB) continuava bastante reduzida.

Traz o **modo real** e o **modo protegido**.

Modo Protegido

Programas estão separados e protegidos uns dos outros por barreiras para que não colidam entre si.

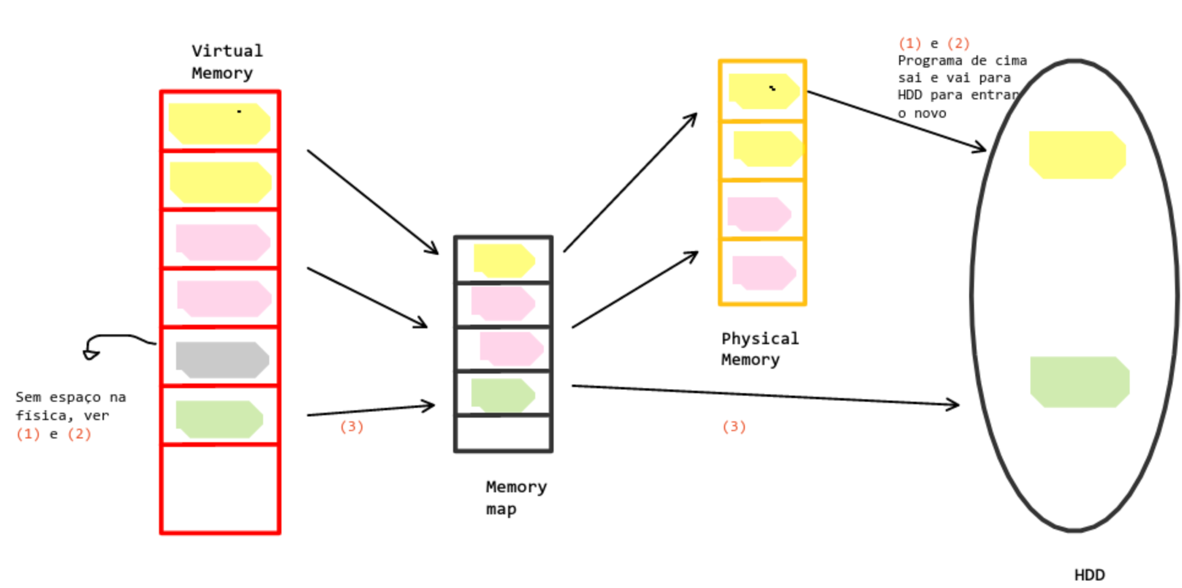
O 8086 não tinha este modo.

Modo Real

Processadores possuem compatibilidade como se fossem um 8086.

Foi nesta altura que surgiu a **memória virtual**.

Memória Virtual



Fica no **memory map** porque a memória física se encontra ocupada.

Apenas vai para a **memória física** quando outra slot ficar disponível.

Memory Map

Tabela com tantas entradas como a da memória virtual (nº páginas).

Ajuda a guardar a ordem das páginas. As alterações de ordem apenas começam a aparecer a partir daí.

Nota:

O disco é 1000x mais lento que a RAM.

O endereço virtual tem mais bits que o endereço real.

Endereços virtuais são visíveis pela unidade de memoria.

MMU

Periférico de hardware, tem a responsabilidade de traduzir endereços lógicos em endereços físicos.

80386

Primeiro processador a vir com registos de 32 bits.

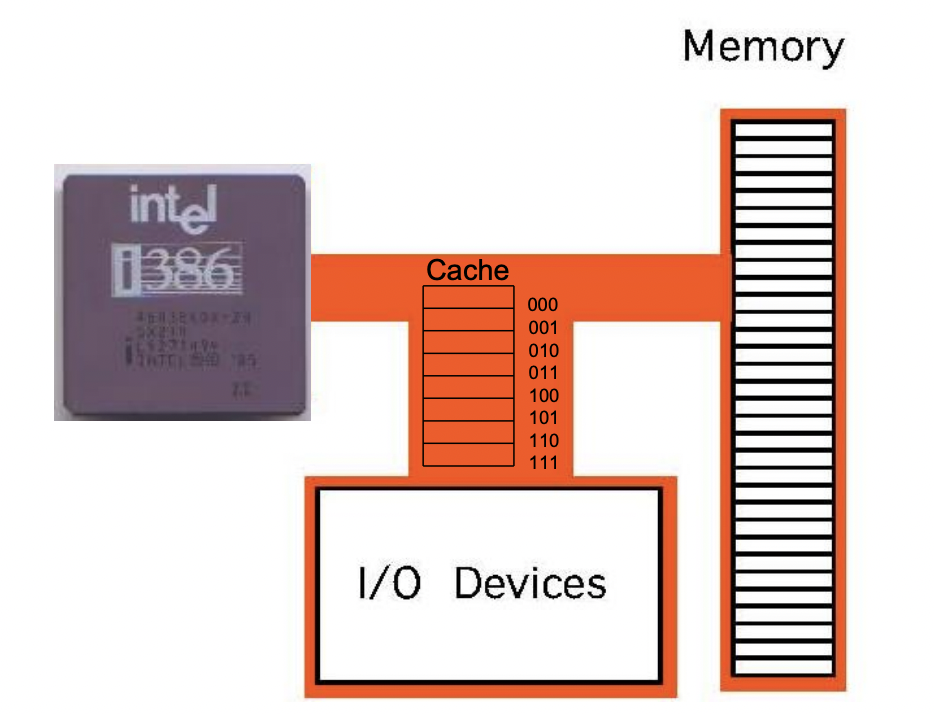
Introdução da **memória cache** (16 bytes).

Pre-fetch Cache Memory

Cache -> guarda a informação em flip-flops.

Nota:

+- 100 vezes mais rápida que a memória convencional.



Nesta fase apenas é usada para guardar instruções.

Vai buscar várias posições na memória e guarda na cache para sem executadas sequencialmente.

Diferença entre a Cache Nível 1 e Nível 2

**Qual a localização da cache consoante o seu nível?\_**

A cache de nível 1 está sempre dentro do processador.

A cache de nível 2 pode estar dentro do processador ou não.

**Qual a mais rápida?**

A cache de nível 1 é mais rápida que a de nível 2.

**Quantos níveis de cache podem existir?**

Ainda pode existir mais um nível de cache, ou seja, 3 níveis no total.

**Qual a relação entre a distância da cache ao CPU?**

Quanto mais nos afastamos da CPU, a cache é mais lenta, no entanto contém mais capacidade de armazenamento.

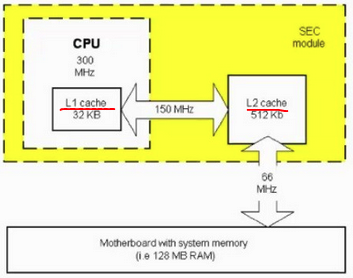
**Como se vai buscar informação na cache?**

Basicamente o processador começa por procurar o programa na cache L1 e se o encontrar, obtém-no muito rapidamente.

A partir do momento que parte para outras caches (ou até mesmo se chegar a ir até à RAM ou disco), a rapidez diminui.

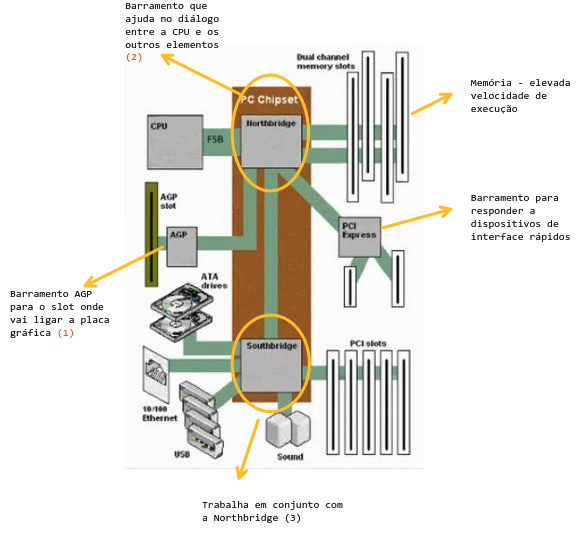
**Qual o motivo de criar várias caches ao invés de apenas 1?**

Optam por fazer várias caches em vez de uma grande dentro da CPU, porque iria ser ocupado muito espaço lá dentro.



Barramentos

Arquitetura com diferentes níveis de velocidade



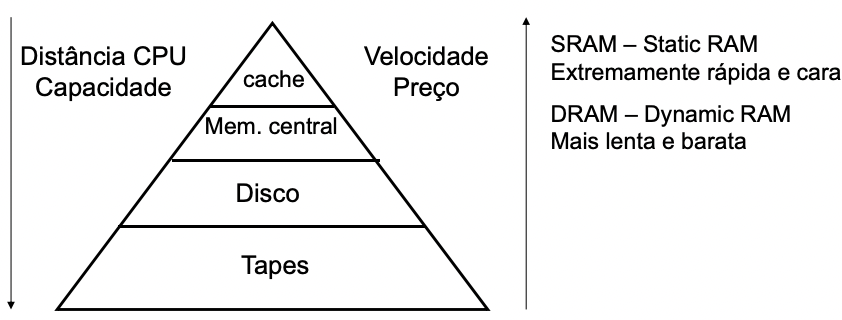
1. Devido ao investimento na multimédia (MMX) que exige transferências rápidas de informação.
2. No início existia um BUS que fazia a ligação entre a CPU e os vários elementos. Com a evolução dos diferentes dispositivos, e consequentemente com a discrepância de velocidades entre eles, foi necessário um dispositivo que ajudasse a dividir o bus de dados pelas respetivas velocidades. Entregava assim as velocidades corretas a cada elemento com a ajuda de outros barramentos.
3. Em conjunto com a NorthBridge, gere o bus de dados para os dispositivos mais lentos (os que vemos em baixo).

Hierarquia de Memória

Conceito:

Dotar a máquina de vários níveis de memória, tão mais rápidos (e com menor capacidade) quanto mais perto se encontrarem do processador.

Cada nível contém uma cópia do código e dados mais usados em cada instante.



Inclusão

A cache irá conter instruções, no entanto essas instruções são uma réplica de um pedaço da memória central.

A própria memória central é uma replica da memória virtual.

Pergunta de Exame

**Para que é necessária a cache?**

Como explicado anteriormente, a cache tem o objetivo de melhorar a velocidade de execução do processador, e consequentemente da máquina.

A memória principal é mais lenta que a cache, portanto ao existirem alguns programas na cache, conseguimos ir buscá-los e processá-los muito mais rapidamente, tendo melhor performance de execução de programas.

Atenção

Não responder que serve para aumentar o armazenamento. Aliás, a cache até tem pouco espaço de armazenamento.

Proximidade

O princípio da proximidade, permite acelerar os acessos à mesma hierarquia.

**Temporal**

Guarda em cache os elementos que foram acedidos há menos tempo (recentemente).

É lento na 1ª vez que está a ser lido, porém depois que é guardado na cache, a 2ª vez já é mais rápida.

**Espacial**

Fica-se na proximidade dos endereços relativos ao endereço que foi executado no momento.

Porém na **proximidade espacial**, não existe uma taxa de acerto de 100% para obter o programa.

Como se fica na proximidade dos endereços, o processo é tentativa-erro.

Terminologias

* Linha
* Bloco
* Hit
* Miss

Objetivo:

Aumentar o **HitRate** (diminuir o **MissRate**, nº de Miss)

Mapeamento Direto

Os endereços de memória são associados aos endereços de cache com a mesma tecnologia (os **n bits finais** são iguais).

Vai gerar conflitos quando outros blocos de memória com a mesma terminação forem colocados na cache e já existir alguma coisa guardada em memória.

|  |
| --- |
| 000 |
| 001 |
|  |
|  |
|  |
|  |
|  |

|  |
| --- |
| 00000 |
| 00001 |
| ... |
| 01000 |
|  |
|  |
|  |

**n bits finais iguais**, e já existe um endereço a ser ocupado com esses bits.

**Solução:**

Colocar a tag (que neste caso são os 2 primeiros bits)

|  |
| --- |
| 00000 |
| ... |
| 01000 |

|  |  |
| --- | --- |
| 00 | 000 |
| ... | ... |
| 01 | 000 |

**TAG**

Os restantes bits do endereço são lá colocados.

**Valid Bits**

CPU consegue saber se o programa que está na cache é valido com um bit extra que indica se os dados dessa linha são válidos.

Uma imagem com texto

Descrição gerada automaticamente

Questão

Considere uma máquina com um espaço de endereçamento de 32 bits, uma cache de 64KB, mapeamento direto e blocos de 1 byte.

Quantos bits são necessários para a tag?

1 bit \* 64Kbits 128KB 64KB

VB TAG CACHE

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

16 bits 8 bits

k ⬄ 1024

32 bits

1000 ... 000

N = 32

4 \* 1GB

8KB = 64K bits

TAG = 32 – nº bits End. TAG

⬄ TAG = 32 – 16 = 16

RAM

|  |
| --- |
|  |
| ... |
|  |
|  |

N

**Capacidade Total da Cache**, contando com os **bits da tag + valid bits**?

Capacidade Total = 128KB + 64KB + 8KB = 200KB

1. **Quantos bits são necessários para a TAG?**

Uma imagem com texto

Descrição gerada automaticamente

1. **Capacidade Total da Cache?**

Uma imagem com mesa

Descrição gerada automaticamente

Arquiteturas Super-Escalares

Inicialmente o processador tratava de um programa de cada vez e outros tinham de ficar à espera – elevado tempo de execução.

Solução (Pipeline)

Fazer uma instrução de um programa e quando essa instrução acabar, fazer a mesma instrução do programa seguinte, ao mesmo tempo o primeiro programa continua para a instrução seguinte, repetindo-se esse processo.

Espécie de produção em série.

Para isto foi necessário mudar a unidade de controlo para começar a gerir vários programas ao mesmo tempo.

Tipos de Pipeline

Pipeline Simples

Processador realiza até 4 instruções diferentes, de programas distintos, em simultâneo.

Super-Pipeline

Processador realiza várias instruções iguais de programas diferentes ao mesmo tempo.

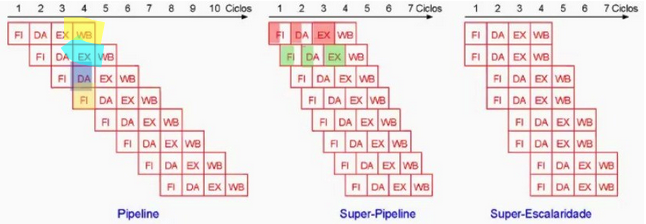
Tem um desfasamento no início do arranque de cada operação.

Antes de acabar o fetch (ir buscar à memória) da primeira instrução, começa o fetch da segunda instrução e assim sucessivamente.

Super-Escalaridade

* Aqui temos 2 ou mais instruções a começar ao mesmo tempo.
* Isto devido à elevada capacidade da CPU.
* Parecido ao anterior, mas com mais capacidade de processamento.

Nas Super, o microprocessador é quem determina quantas e quais instruções se fazem ao mesmo tempo.



Limitações das Super

* As instruções podem ocorrer sem outras anteriores terem sido finalizadas.
* Essas instruções podem precisar de dados das anteriores, gerando aí um problema, pois precisam de esperar por elas, **travando todo o processo**.

“True Data Dependency”

Tem que se esperar o resultado de uma operação para se poder executar outra.

Uma imagem com mesa

Descrição gerada automaticamente

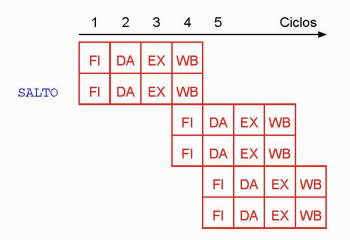
“Procedural Dependency”

Saltos são sempre um problema!

Nota:

Nos **RISC**, as instruções têm uma dimensão fixa.

Se for **CISC** ainda se acrescenta o problema de controlar a dimensão da instrução.



Solução

Colocar essas instruções, que necessitam de outras, para último e assim as que estavam a ficar atrasadas podem ser processadas.

O próprio processador é quem gere isto.

Porém, isto pode gerar outro problema

Visto que iremos fazer saltar uma instrução para o fim e estamos a dar liberdade a outras, essas outras podiam precisar da que saltou ou não realmente necessárias de serem realizadas visto que a que saltou podia mostrar que já não era preciso realizar mais.

Arquitetura Híbrida

CISC (Complex Instruction Set Computer)

* Consegue incorporar vários softwares (principalmente com a ideia dos mais antigos) no mesmo processador.
* Contudo, os processos ficavam lentos porque a linguagem era muito mais pesada e com milhares de instruções.
* Instruções efetuadas sobre a memoria, retirando pouco partida da sua localização espacial.

RISC (Reduced Instruction Set Computer)

* Processador extremamente simples, operações feitas mais no processador, não sendo necessário ir tanto à memoria.
* Portanto, grande rapidez na execução de tarefas.
* Instruções de formato fixo, cujo código tem sempre o mesmo tamanho, o que lhe permite a descodificação da instrução e trabalhar com pipelining.

Nota:

Os processadores INTEL ainda usam o CISC.

No entanto o RISC é usado em telemóveis, consolas de jogos, ...

O RISC é mais eficiente que o CISC – Pipeline também mais adaptado ao RISC.

Estas tecnologias colidiam um pouco com o Pipeline, pois tinham muitas instruções diversificadas, levando a que uma instrução grande atrasasse todo o processo do pipeline.

Junção das Duas

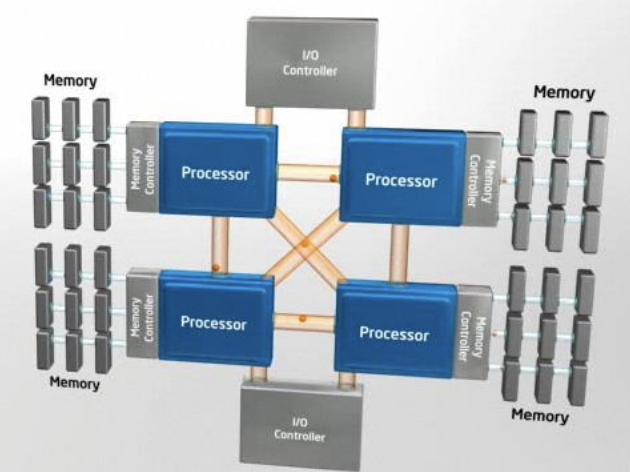
Como a Intel sempre se interessou em ter softwares antigos disponíveis, trouxe a arquitetura híbrida que traz um **núcleo RISC** e um **descodificador CISC**.

FSB

* Consiste em um **barramento compartilhado**, que liga o processador ao chipset.
* Como é usado não apenas para a comunicação entre os núcleos do processador e a memória, mas também para a comunicação entre os 2 ou 4 núcleos do processador, acaba por estrangular o acesso à memória, prejudicando o desempenho do sistema.
* Funciona bem até 3 ou 4 processadores.
* Alterado a partir do processador i3, passou a ser QuickPath.

QuickPath

* Agora cada núcleo passa a ter um **controlador de memória**, porém a memória é a mesma.
* Passa a haver um canal de comunicação (link) entre os núcleos.



Exemplo com 4 núcleos

Cada **link** é full-duplex, passam informação para ambos os lados.

Discos Rígidos

**Linhas vermelhas:** Pistas

**Linhas azuis:** Marcam os setores (fatias pizza), suportam 512Bytes

Possível Questão de Exame

Uma unidade de disco com **n pratos** (2n faces) terá **2n cabeças e 2n pistas por cilindro.**

Exemplo com 2 pratos:

Capacidade de uma pista = Nº de Setores por Pista \* Nº de Bytes por Setor

= 8 \* 512B = 4096B

Capacidade de um cilindro = Nº de Cabeças \* Capacidade de uma pista

= 4 \* 4096B

= 16384

= 16KB

Refrescamento de Memória – RAM

Possível Questão de Exame

* Refrescamento por “Burst”
* Refrescamento Distribuído
* Refrescamento Transparente

O que caracteriza cada um destes refrescamentos?

Refrescamento por “Burst”

De 2 em 2 ms (milissegundos) refresca um conjunto de linhas (n). Durante esse tempo de refrescamento, a RAM não pode estar em funcionamento, o que pode prejudicar na hora de a utilizar para tarefas importantes que o processador solicite.

Refrescamento Distribuído

Tem muitas paragens, porém devido a essas paragens acaba por cooperar mais com o processador.

Refrescamento Transparente

Mais eficiente.

Deteta períodos em que o acesso à memória se encontra parado e efetua o refrescamento.