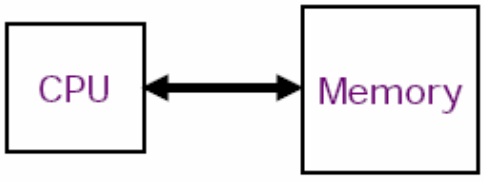
Aula 16: Memória Principal e Memória Virtual

6.823 L7- 2
Krste Asanovic

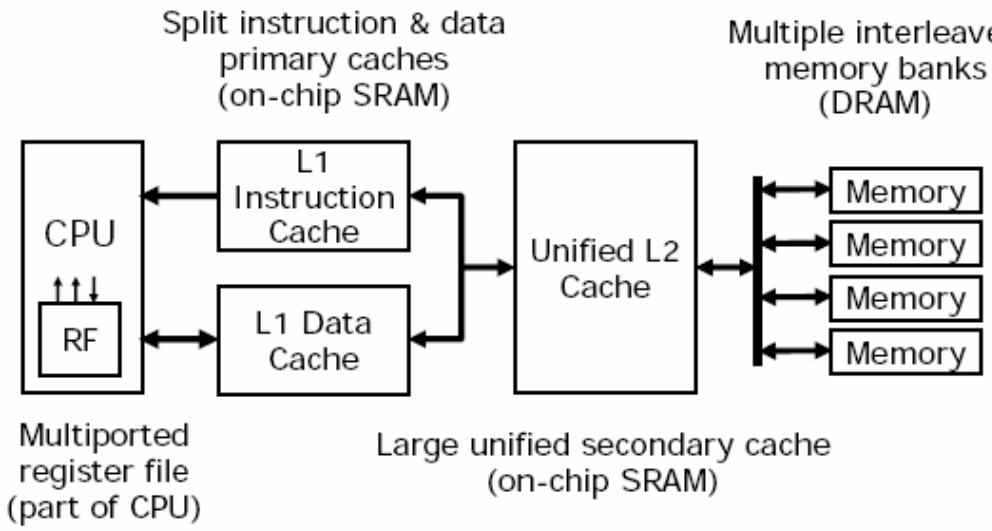
CPU-Memory Bottleneck



Performance of high-speed computers is usually limited by memory *bandwidth* & *latency*

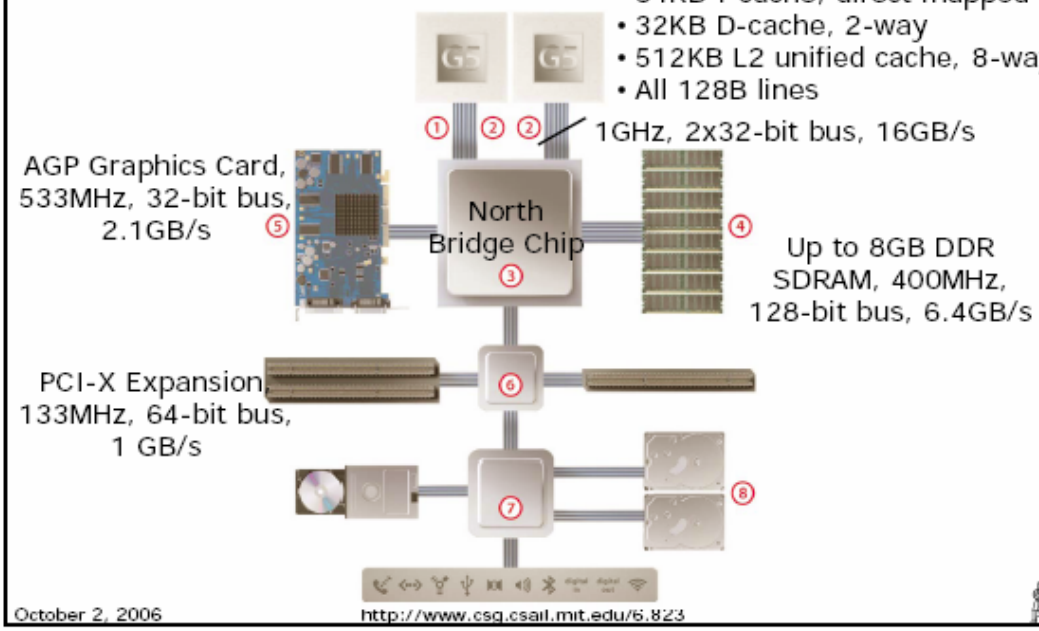
- Latency (time for a single access)
  Memory access time >> Processor cycle time
- Bandwidth (number of accesses per unit time)
  if fraction *m* of instructions access memory,
  ⇒1+*m* memory references / instruction
  ⇒ CPI = 1 requires 1+*m* memory refs / cycle

A Typical Memory Hierarchy c.2006



Workstation Memory System (Apple PowerMac G5, 2003)

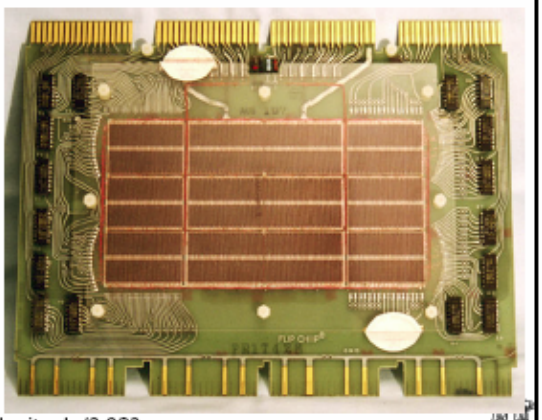
Dual 2GHz processors, each has:
- 64KB I-cache, direct mapped
- 32KB D-cache, 2-way
- 512KB L2 unified cache, 8-way
- All 128B lines



Core Memory

- Core memory was first large scale reliable main memory
  - invented by Forrester in late 40s at MIT for Whirlwind project
- Bits stored as magnetization polarity on small ferrite cores threaded onto 2 dimensional grid of wires
- Coincident current pulses on X and Y wires would write cell and also sense original state (destructive reads)

- Robust, non-volatile storage
- Used on space shuttle computers until recently
- Cores threaded onto wires by hand (25 billion a year at peak production)
- Core access time ~ 1μs

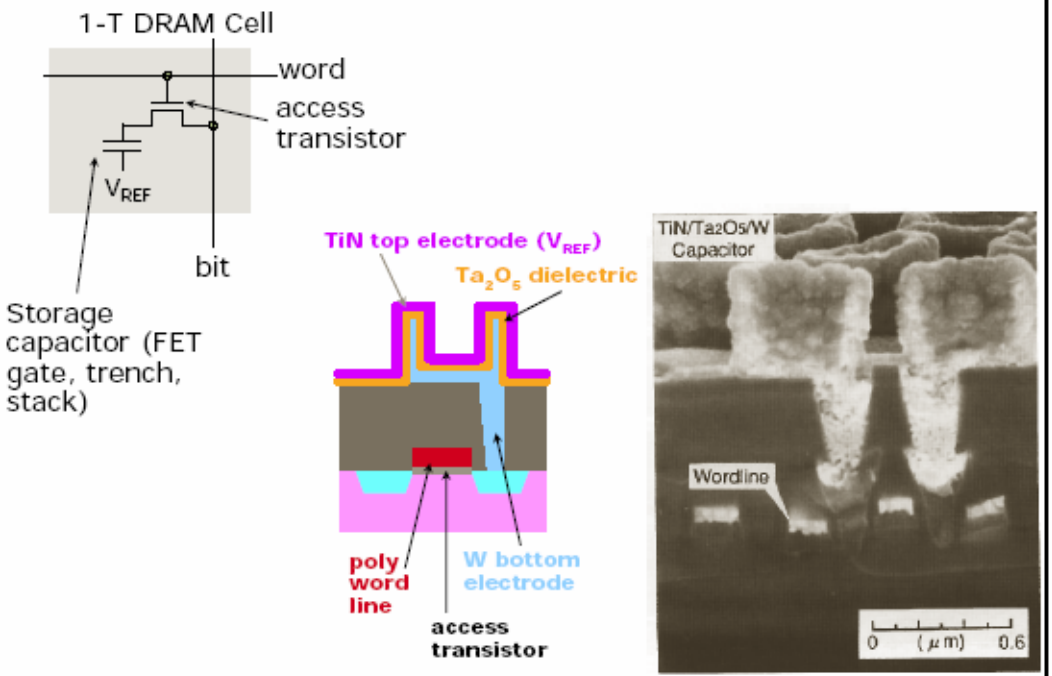


DEC PDP-8/E Board,
4K words x 12 bits, (1968)

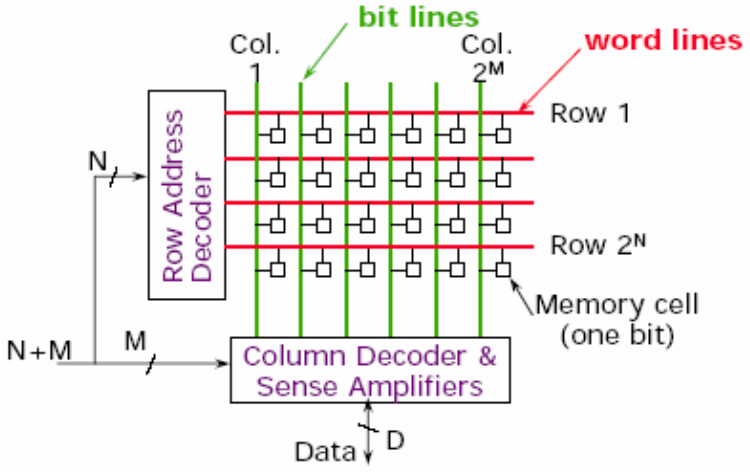
Semiconductor Memory, DRAM

- Semiconductor memory began to be competitive in early 1970s
  - Intel formed to exploit market for semiconductor memory
- First commercial DRAM was Intel 1103
  - 1Kbit of storage on single chip
  - charge on a capacitor used to hold value
- Semiconductor memory quickly replaced core in 1970s

One Transistor Dynamic RAM

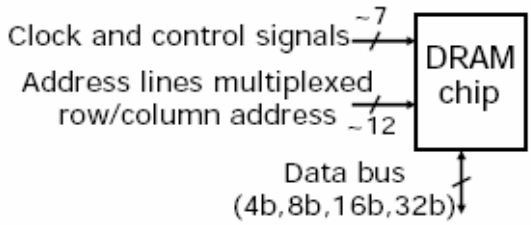


DRAM Architecture



- Bits stored in 2-dimensional arrays on chip
- Modern chips have around 4 logical banks on each chip
  - each logical bank physically implemented as many smaller arrays

DRAM Packaging



- DIMM (Dual Inline Memory Module) contains multiple chips with clock/control/address signals connected in parallel (sometimes need buffers to drive signals to all chips)
- Data pins work together to return wide word (e.g., 64-bit data bus using 16x4-bit parts)


72-pin SO DIMM


168-pin DIMM

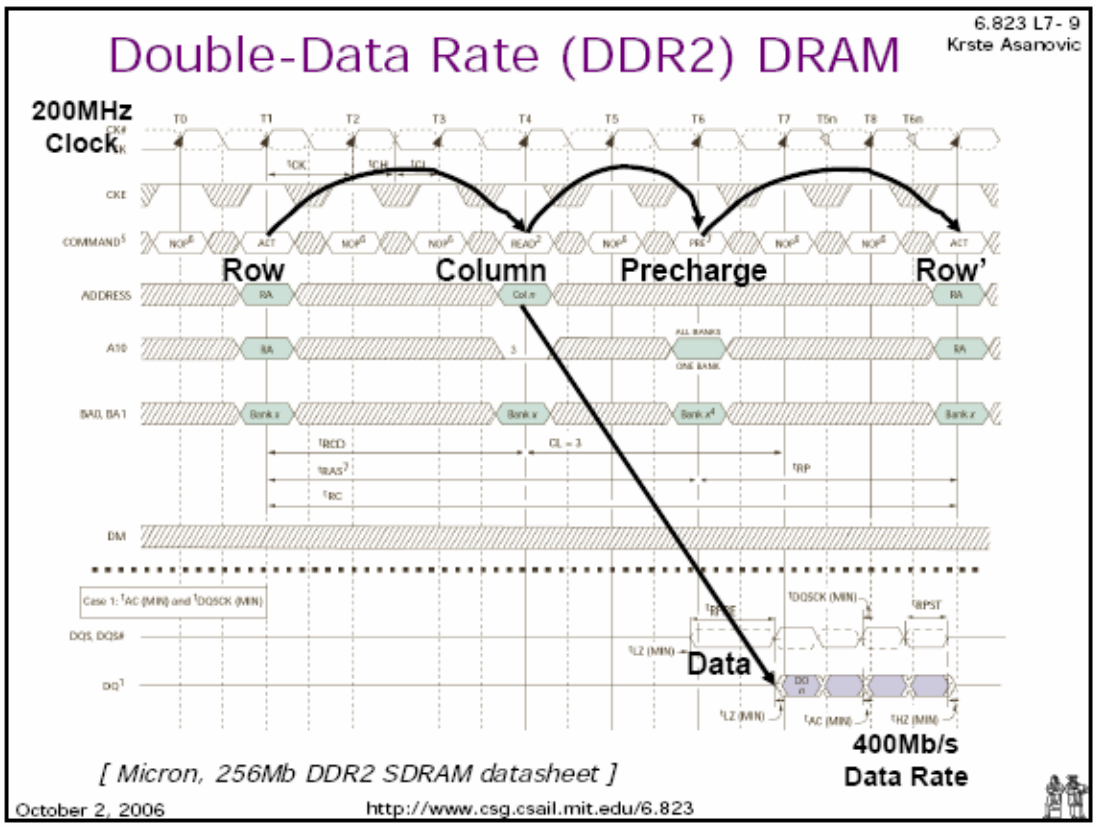
DRAM Operation

Three steps in read/write access to a given bank

- Row access (RAS)
  - decode row address, enable addressed row (often multiple Kb in row)
  - bitlines share charge with storage cell
  - small change in voltage detected by sense amplifiers which latch whole row of bits
  - sense amplifiers drive bitlines full rail to recharge storage cells
- Column access (CAS)
  - decode column address to select small number of sense amplifier latches (4, 8, 16, or 32 bits depending on DRAM package)
  - on read, send latched bits out to chip pins
  - on write, change sense amplifier latches which then charge storage cells to required value
  - can perform multiple column accesses on same row without another row access (burst mode)
- Precharge
  - charges bit lines to known value, required before next row access

Each step has a latency of around 20ns in modern DRAMs

Various DRAM standards (DDR, RDRAM) have different ways of encoding the signals for transmission to the DRAM, but all share same core architecture

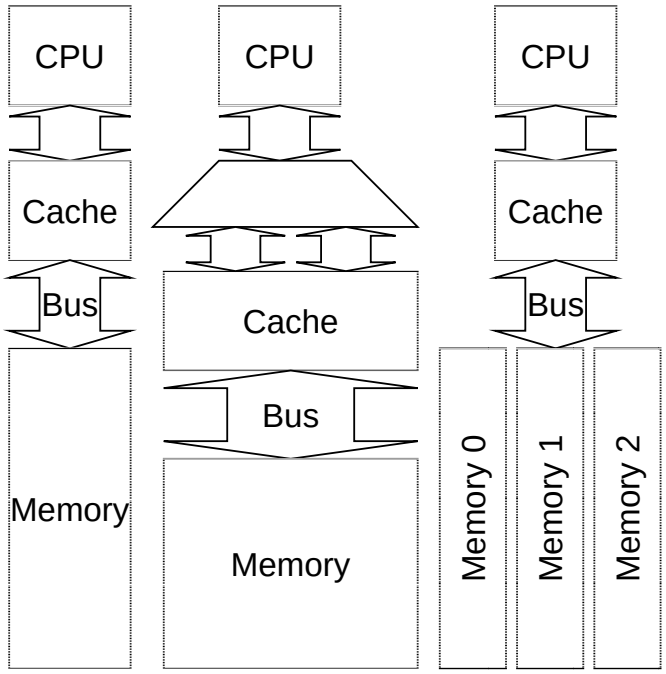


Memória Principal

- Performance na memória principal:
  - Latência: Miss Penalty na Cache
    - *Access Time*: tempo entre requisição e retorno de palavra
    - *Cycle Time*: tempo entre requisições
  - Bandwidth: transferência de dados em bits/seg.
    - Influencia I/O & e Miss Penalty de Caches com blocos grandes (L2)
- Memória principal é implementada em *DRAM*: *Dynamic Random Access Memory*
  - Dinâmica == precisa ser refrescada periodicamente (8 ms)
  - Endereços divididos em duas partes (Memória é uma matrix 2D):
    - *RAS* ou *Row Access Strobe*
    - *CAS* ou *Column Access Strobe*
- Cache usam *SRAM*: *Static Random Access Memory*
  - Não precisam ser refrescadas (6 transistores/bit vs. 1 transistor
    Tamanho: DRAM/SRAM - *4-8*,
    Custo/Cycle Time: SRAM/DRAM - *8-16*

Performance de Memória Principal

- *Simples*:
  - CPU, Cache, Barramento, Memória de mesma largura (32 bits)
- *Larga*:
  - CPU/Mux 1 palavra; Mux/Cache, Barramento, Memória N palavras (Alpha: 64 bits & 256 bits)
- *Intercalada*:
  - CPU, Cache, Barramento 1 palavra: Memória N Módulos (4 Módulos); exemplo possui intercalação de palavras



Performance de Memória Principal

- Temporização
  - 1 ciclo para enviar endereço,
  - 6 ciclos para acess time, 1 ciclo para enviar dados
  - Bloco da cache tem 4 palavras

- *Simples M.P.* = 4 x (1+6+1) = 32
- *Larga M.P.* = 1 + 6 + 1 = 8
- *Intercalada M.P.* = 1 + 6 + 4x1 = 11

Bancos de Memória Independentes

- Permite acessos independentes
  - Multiprocessador
  - I/O
  - Miss under Miss, caches não-blocantes
- *Superbanco*: Toda memória ativa durante a transferência de um bloco
- *Banco*: parte do superbanco que é intercalado

Bancos de Memória Independentes

- Quantos bancos?

  Número de bancos = número de ciclos para acessar palavra em banco
- Com o aumento da densidade de DRAM => menor número de chips => mais difícil de se ter bancos

Evitando Conflitos em Bancos

- Muitos bancos

```
int x[256][512];
    for (j = 0; j < 512; j = j+1)
          for (i = 0; i < 256; i = i+1)
                x[i][j] = 2 * x[i][j];
```

- Até mesmo com 128 bancos haverão conflitos, já que 512 é um múltiplo de 128
- SW: mudança da ordem de acesso aos elementos do loop
- HW: Número primo de bancos
  - número do banco = endereço mod número do banco
  - endereço dentro do banco = endereço / número de bancos
  - módulo & divisão por acesso de memória?
  - endereço dentro do banco => use número primo == 2^N - 1 (3,7,31)
  - número de palavras em um banco é potência de dois

Cálculo do Número do Banco

- *Chinese Remainder Theorem*

  Se dois conjuntos de inteiros ai e bi seguem estas regras

  $b_i = x \bmod a_i, 0 \le b_i < a_i, 0 \le x < a_0 \times a_1 \times a_2 \times \ldots$

  e ai e aj são co-primos se i != j, então inteiro x possui somente um único mapeamento:
  - número do banco = b_0, número de bancos = a_0 (= 3 no exemplo)
  - endereço no banco = b_1, número de palavras no banco = a_1 (= 8 no exemplo)
  - endereço é dado por par (b_1, b_0)

		Seq. Interleaved			Modulo Interleaved		
Bank Number:		**0**	**1**	**2**	**0**	**1**	**2**
	Address						**within**
Bank:	***0***	**0**	**1**	**2**	**0**	**16**	**8**
	1	**3**	**4**	**5**	**9**	**1**	**17**
	2	**6**	**7**	**8**	**18**	**10**	**2**
	3	**9**	**10**	**11**	**3**	**19**	**11**
	4	**12**	**13**	**14**	**12**	**4**	**20**
	5	**15**	**16**	**17**	**21**	**13**	**5**
	6	**18**	**19**	**20**	**6**	**22**	**14**
	7	**21**	**22**	**23**	**15**	**7**	**23**

Intercalamento Específico para DRAM

- Múltiplos acessos RAS: único RAS, diversos CAS (page mode)
  - 64 Mbit DRAM: cycle time = 100 ns, page mode = 20 ns
- Novas DRAMs para redução do tempo
  - *Synchronous DRAM*: Transferências síncronas com clock
  - *RAMBUS*: reinventou a interface p/ DRAMs
    - cada chip é um módulo vs. parte da memória
    - faz próprio refrescamento
    - quantidade de bytes retornados é variável
    - 1 byte / 2 ns (500 MB/s por chip)
- Outros tipos de memória?
  - VRAM: dois portos (1 para leitura, 1 para escrita)

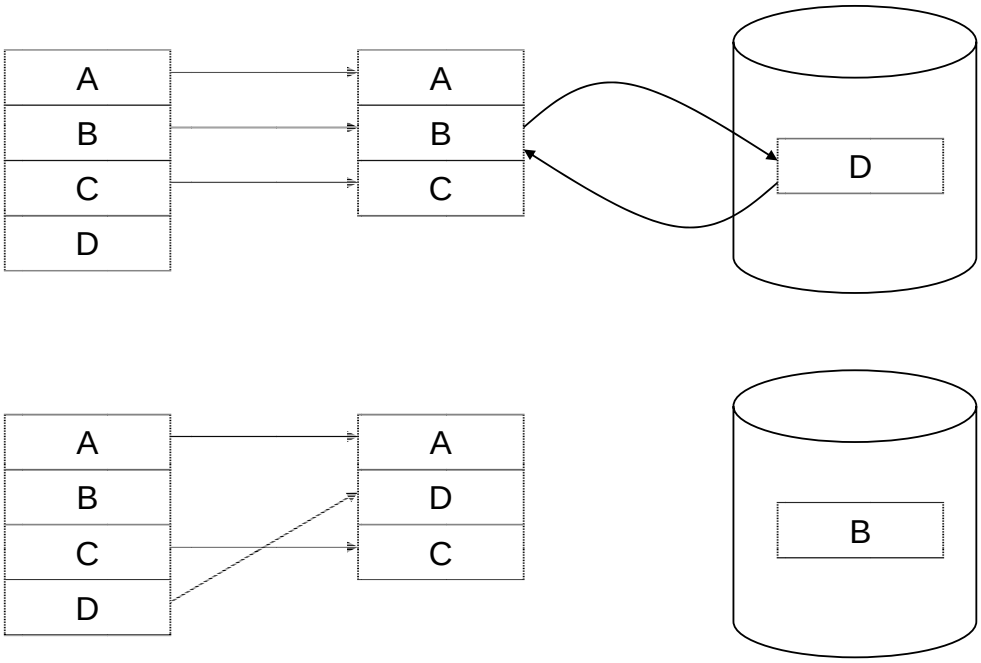
Memória Virtual

- Histórico
- Endereço virtual (2^{32}, 2^{64}) para endereço físico (2^{28})
- Termos equivalentes a caches:
  - Bloco?
  - Miss?
- Como memória virtual é diferente de caches?
  - 4Qs

Memória Virtual
Histórico

- Espaço de memória principal limitado
  - Programas grandes eram limitados por overlays, que eram utilizados para carregar partes de programas
    - Controlados pelo usuário
- Sistema com múltiplos processos
  - Código deveria ser independente da posição de carga

Memória Virtual



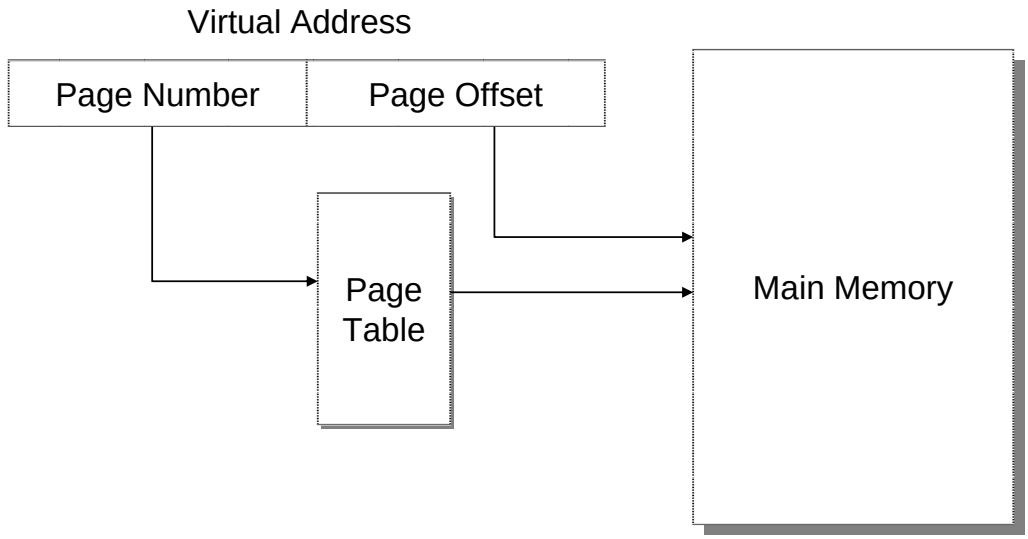
Memória Virtual vs. Caches

Parameter	First-level Cache	Virtual Memory
Block (page) size	16-128 bytes	4K-64K bytes
Hit time	1-2 cycles	40-100 cycles
Miss penalty	8-100 cycles	700K-6M cycles
Miss rate	0.5-10%	0.00001-0.001%
Data memory size	0.016-1MB	16-8192MB

Memória Virtual

- 4Qs para VM?
  - Q1: Onde bloco pode ser colocado no nível mais alto?
    Fully associative
  - Q2: Como bloco é encontrado no nível mais alto?
    Page table + TLB
  - Q3: Qual bloco deve ser trocado em um miss?
    LRU
  - Q4: O que acontece em uma escrita?
    Write back

Tradução Rápida



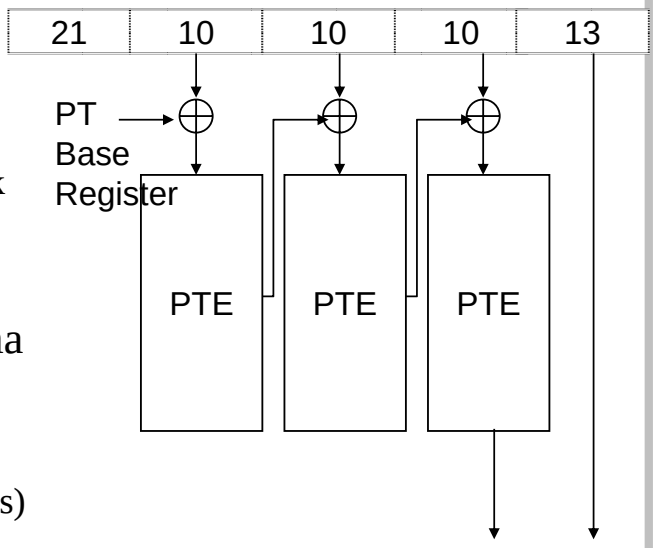
1 acesso de memória == 2 acessos (1 p/ Page Table, 1 p/ busca de dado)
Solução: TLB (*cache* para acessos mais recentes à Page Table)

Selecionando Tamanho de Página

- Páginas maiores
  - Page table tem tamanho inversamente prop. ao tamanho da página
  - Transferências de uma página grande para disco é melhor
  - Número de entradas na TLB é limitada pelo clock, portanto, tamanho de página maior aumenta localidade capturada pela TLB
- Páginas menores
  - Fragmentação
- Solução híbrida: tamanhos múltiplos
  - Alpha: 8KB, 16KB, 32 KB, 64 KB

VM no Alpha

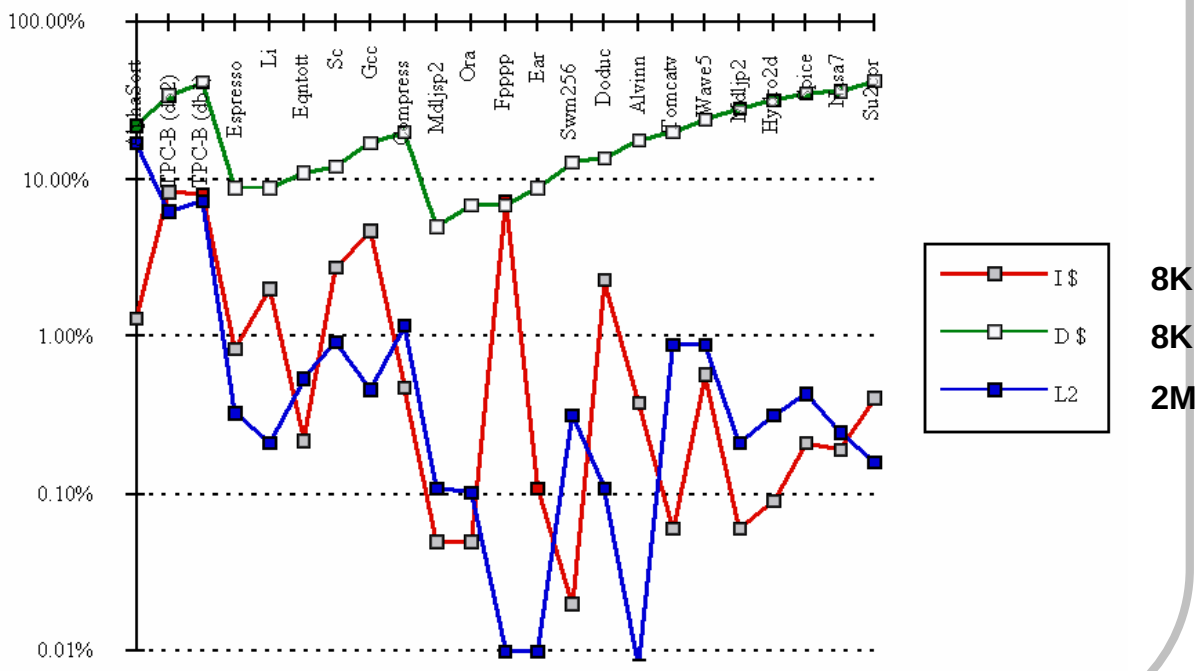
- "64-bit" de endereço divididos em 3 segmentos
  - seg0 (bit 63=0) user code/heap
  - seg1 (bit 63 = 1, 62 = 1) user stack
  - kseg (bit 63 = 1, 62 = 0) kernel
- Page table em 3 níveis, cada uma com uma página
  - Alpha com 43 bits para VA
  - (mas permite expansão para 55 bits)
- PTE bits; valid, kernel & user read & write enable (Não possui dirty bit)



Alpha 21064

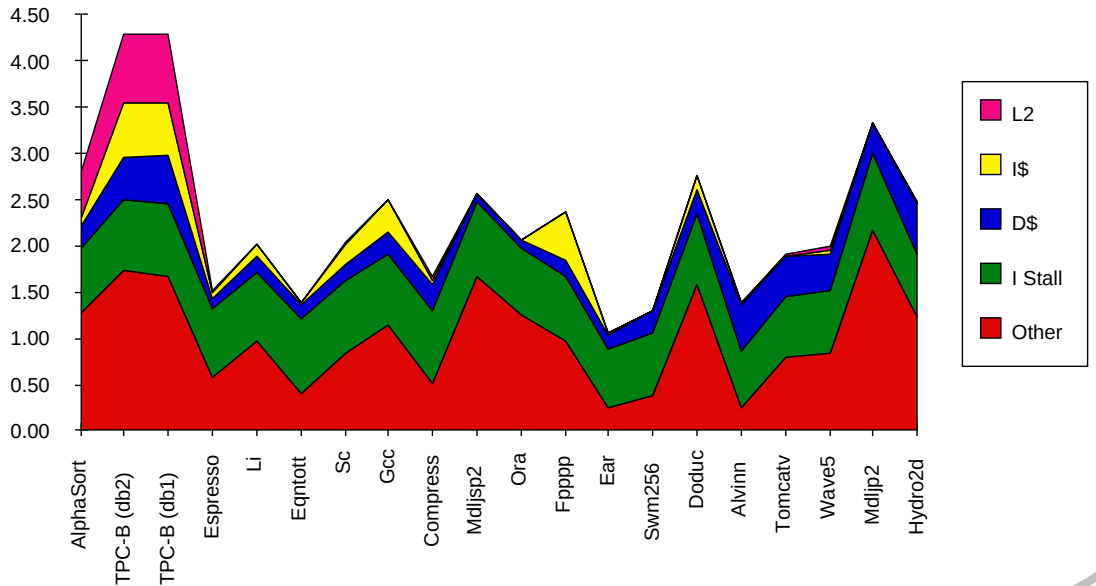
- Separate Instr & Data TLB & Caches
- TLBs fully associative
- TLB updates in SW ("Priv Arch Libr")
- Caches 8KB direct mapped
- Critical 8 bytes first
- Prefetch instr. stream buffer
- 2 MB L2 cache, direct mapped (off-chip)
- 256 bit path to main memory, 4 x 64-bit modules

Performance de Memória do Alpha: Miss Rates



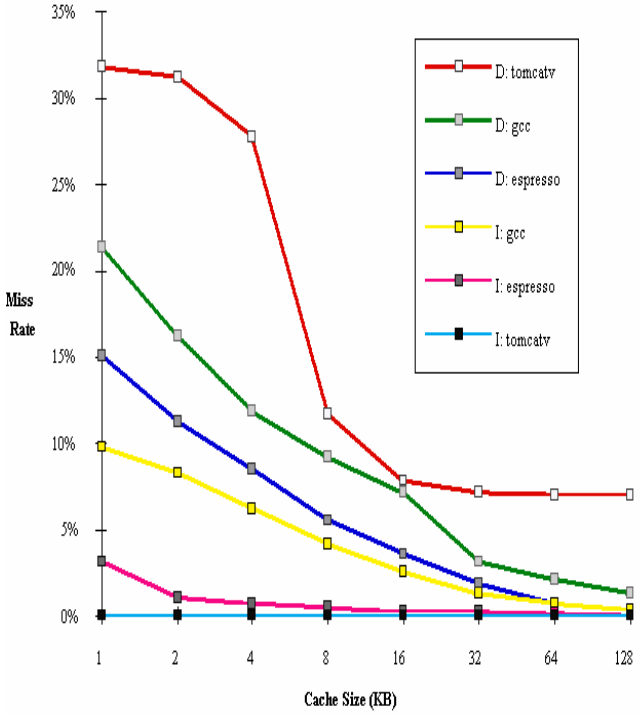
Alpha CPI Components

- I Stall: previsão errada de branches;
  Outros: hazards estruturais e de dados



Previsão de Performance de Caches a Partir de Prg Diferente (ISA, compilador,…)

- MR para cache de dados de 4KB é 8%, 12%, ou 28%?
- MR para cache de instr. de 1KB é 0%, 3%, ou 10%?
- MR para cache de dados de 8KB da Alpha vs. MIPS: 17% vs. 10%



Simulando “Trace” Pequeno