### CHƯƠNG 7: COMPUTER ORGANISATION

#### \*Tất cả các thuật ngữ viết tắt:

1. CPU - Central Processing Unit
2. RAM - Random Access Memory
3. ROM - Read-Only Memory
4. I/O - Input - Output
5. ALU - Arithmetic Logic Unit
6. DMA - Direct Memory Access
7. GPU - Graphics Processing Unit
8. BIOS - Basic Input/Output System
9. Cache - Temporary memory storage for frequently accessed data
10. IDE - Integrated Development Environment
11. IP - Internet Protocol
12. FBS - Font-Side Bus
13. PCI - Peripheral Component Interconnect

\* Định luật Moore

- Đề xuất năm 1965

- Định luật nói rằng số lượng transistor trên một hệ thống vi xử lý (integrated circuit) sẽ tăng gấp đôi mỗi 18-24 tháng.

\* Kiến trúc Von Neumann

- Kiến trúc này xác định cách mà một máy tính có thể thực hiện các tác vụ bằng cách chia máy tính thành hai phần chính: bộ nhớ và vi xử lý (Central Processing Unit - CPU).

- Kiến trúc Von Neumann bao gồm các yếu tố sau:

1. Bộ nhớ chung (main memory): Lưu trữ chương trình và dữ liệu đang được sử dụng.
2. Bộ vi xử lý (CPU): Thực hiện các tính toán và chuyển đổi các lệnh từ chương trình.
3. Đầu vào/đầu ra (I/O): Cung cấp các thiết bị nhập liệu và xuất liệu cho máy tính.
4. Bus: Là một dãy dây để truyền dữ liệu giữa các thành phần của máy tính.

\*Định luật Amdahl

- Định luật Amdahl cho biết rằng tốc độ tăng tối đa của một hệ thống máy tính phụ thuộc vào phần của chương trình có thể được cải thiện bằng việc sử dụng nhiều vi xử lý hơn.

- Công thức cụ thể của Định luật Amdahl là:

Speedup = 1 / ( (1 - P) + (P / S) )

Trong đó:

• P là tỷ lệ phần của hệ thống được nâng cấp.

• S là tốc độ tăng trưởng của phần được nâng cấp so với phần cũ.

- Ví dụ: Gần đây, một công ty đã nâng cấp một phần của hệ thống máy tính của họ để tăng tốc độ xử lý. Tỷ lệ phần mới nâng cấp là 40% và tốc độ tăng trưởng của phần mới nâng cấp là 2 lần so với phần cũ. Tính tốc độ tăng trưởng tối ưu theo Định luật Amdahl.

P = 40% = 0.40

S = 2

Speedup = 1 / ( (1 - P) + (P / S) ) = 1 / ( (1 - 0.40) + (0.40 / 2) ) = 1 / (0.60 + 0.20) = 1 / 0.80 = 1.25

Kết quả cho thấy rằng tốc độ tăng trưởng tối ưu của hệ thống sau khi nâng cấp là 1.25 lần so với trước khi nâng cấp.

\* Thành phần của CPU

Thành phần chính của CPU (Central Processing Unit) gồm có:

1. Control Unit - CU (Điều khiển) : Nhiệm vụ chính là điều khiển và quản lý hoạt động của CPU
2. Arithmetic Logic Unit - ALU (Đơn vị tính toán & logic) : Thực hiện các phép tính số học và các phép so sánh logic.
3. Registers (Bộ đăng ký) : Là những vùng nhớ tạm thời để lưu trữ các giá trị tạm thời cần thiết để thực hiện các phép tính và chuyển dữ liệu giữa các bộ phận khác.
4. Cache (Bộ nhớ đệm) : Là một vùng nhớ nhanh để lưu trữ các dữ liệu và lệnh thường xuyên sử dụng.
5. Instruction Register (Bộ đăng ký hướng dẫn) : Lưu trữ các lệnh đang được thực hiện hoặc sắp được thực hiện.
6. Address Bus (Đường truyền địa chỉ) : Truyền địa chỉ của dữ liệu hoặc lệnh đang được yêu cầu.
7. Data Bus (Đường truyền dữ liệu) : Truyền dữ liệu giữa CPU và bộ nhớ hoặc các thiết bị ngoại vi.

\* Các bước trong chu trình thực hiện lệnh

Các bước trong chu trình thực hiện lệnh (Instruction cycle) gồm:

1. Fetch (Lấy): CPU lấy lệnh từ bộ nhớ để xử lý.
2. Decode (Giải mã): CPU giải mã lệnh để hiểu nội dung của lệnh.
3. Execute (Thực hiện): CPU thực hiện các phép toán cần thiết theo lệnh.
4. Store (Lưu): CPU lưu kết quả xử lý vào bộ nhớ.
5. Update PC: CPU cập nhật trỏ đến vị trí lệnh tiếp theo trong bộ nhớ.

Sau khi hoàn tất một chu trình, CPU sẽ tiếp tục quay lại bước 1 để xử lý lệnh tiếp theo.

### CHƯƠNG 8: PERFORMANCE

\* Tính Hệ số tăng tốc (Speed up)

Hệ số tăng tốc (Speed up) là một chỉ số đo lường performance của hệ thống, thiết bị hoặc phần mềm. Nó cho biết tốc độ hoạt động tốt hơn của hệ thống khi sử dụng nhiều tài nguyên hơn so với sử dụng một tài nguyên.

Công thức tính Hệ số tăng tốc (Speed up) là:

**Speed up = Thời gian hoạt động của một tài nguyên / Thời gian hoạt động của nhiều tài nguyên**

Ví dụ: Nếu một tác vụ cần 10 giây để hoàn thành với một tài nguyên và chỉ cần 5 giây để hoàn thành với hai tài nguyên, thì Hệ số tăng tốc sẽ là:

Speed up = 10 giây / 5 giây = 2.

Điều này có nghĩa là chương trình đã tăng tốc gấp đôi khi sử dụng nhiều tài nguyên hơn

\* Tính thời gian của một xung (clock)

Thời gian của một xung (clock) là thời gian giữa hai xung liên tiếp trong chu kỳ hoạt động của CPU. Điều này có nghĩa là thời gian mà CPU phải chờ để nhận được một xung mới. Thời gian xung được đo bằng giây hoặc nano giây.

Công thức tính thời gian của một xung (clock) là:

**Thời gian xung = 1 / Tần số xung (1/f)**

Ví dụ, nếu tần số xung của một CPU là 2 GHz (2 billion xung mỗi giây), thì thời gian của một xung là:

Thời gian xung = 1/2GHz =1 / 2 \* 10^9 = 0.5 nano giây

\*Đơn vị đo hiệu năng của vi xử lý

Có nhiều đơn vị khác nhau để đo hiệu năng của vi xử lý, nhưng các đơn vị phổ biến nhất gồm:

1. MIPS (Millions of Instructions per Second): Số lượng các lệnh mà vi xử lý có thể thực hiện trong một giây.
2. FLOPS (Floating Point Operations Per Second): Số lượng phép toán số thực (floating point) mà vi xử lý có thể thực hiện trong một giây.
3. GHz (Gigahertz): Tần số xung của vi xử lý, đo bằng số lần vi xử lý chạy trong một giây.
4. Instructions per second (IPS) - Số lệnh mà vi xử lý có thể thực hiện trong một giây.

\* Cách tính thời gian thực thi một chương trình

Tính thời gian thực thi một chương trình có thể thực hiện bằng cách sử dụng các công cụ đo thời gian hoặc bằng cách sử dụng các hằng số đã biết trong hệ thống. Các bước cơ bản gồm:

1. Chạy chương trình một lần và ghi lại thời gian bắt đầu và kết thúc.
2. Tính toán thời gian chạy chương trình bằng cách trừ thời gian kết thúc cho thời gian bắt đầu.
3. Lặp lại quá trình này để tìm ra thời gian trung bình của chương trình.

### CHƯƠNG 9: MIPS + CHƯƠNG 10: INSTRUCTION SET ARCHITECTURE

\* Các thuật ngữ viết tắt

1. RISC - Reduced Instruction Set Computing
2. ISA - Instruction Set Architecture
3. MIPS - Microprocessor without Interlocked Pipeline Stages
4. OS - Operating System
5. API - Application Programming Interface

\* Hiểu các câu lệnh trong MIPS

Các câu lệnh trong MIPS bao gồm:

1. add - Cộng hai giá trị và lưu kết quả vào một địa chỉ bộ nhớ.
2. sub - Trừ hai giá trị và lưu kết quả vào một địa chỉ bộ nhớ.
3. and - Phép and bitwise giữa hai giá trị và lưu kết quả vào một địa chỉ bộ nhớ.
4. or - Phép or bitwise giữa hai giá trị và lưu kết quả vào một địa chỉ bộ nhớ.
5. slt - So sánh hai giá trị và lưu kết quả vào một địa chỉ bộ nhớ.
6. beq - Chuyển đến một địa chỉ nếu hai giá trị bằng nhau.
7. bne - Chuyển đến một địa chỉ nếu hai giá trị không bằng nhau.
8. j - Chuyển đến một địa chỉ tuyệt đối.
9. jr - Chuyển đến địa chỉ lưu trong một địa chỉ bộ nhớ.
10. lw - Đọc giá trị từ bộ nhớ và lưu vào một địa chỉ bộ nhớ.
11. sw - Lưu giá trị từ một địa chỉ bộ nhớ vào bộ nhớ.

Đây là một số câu lệnh cơ bản trong MIPS, tuy nhiên có rất nhiều câu lệnh

#### \* Cách lưu các kết quả vào các thanh ghi

Trong MIPS, các kết quả của các phép toán được lưu vào các thanh ghi (register). Có 32 thanh ghi trong MIPS, mỗi thanh ghi có độ dài là 32 bit.

Cách lưu kết quả vào thanh ghi trong MIPS như sau:

1. Sử dụng câu lệnh "move"
2. Sử dụng câu lệnh "add" hoặc "sub"
3. Sử dụng lệnh "sw" (store word).
4. Sử dụng các câu lệnh khác như "and", "or", "xor" v.v.

#### \* Các kiểu câu lệnh Assembly (R-type, I-Type, ....)

Trong MIPS, có 3 loại câu lệnh Assembly chính:

1. R-type: Là kiểu câu lệnh dùng cho các phép toán hai vào hai ra (two operand), ví dụ như cộng, trừ, nhân, chia, v.v. Mỗi câu lệnh R-type gồm có 6 trường: opcode, rs, rt, rd, shamt, funct.

Ví dụ: add $s0, $s1, $s2

Trong đó, add là ký hiệu opcode, $s0, $s1, $s2 là 3 thanh ghi operand.

1. I-type: Là kiểu câu lệnh dùng cho các phép toán một vào hai ra (one operand) hoặc phép toán tải dữ liệu (load) hoặc lưu dữ liệu (store). Mỗi câu lệnh I-type gồm có 4 trường: opcode, rs, rt, offset.

Ví dụ: addi $s0, $s1, 100

Trong đó, addi là ký hiệu opcode, $s0 và $s1 là 2 thanh ghi operand, 100 là số nguyên (immediate).

1. J-type: Là kiểu câu lệnh dùng cho các chuyển động (jump) đến một vị trí trong chương trình. Mỗi câu lệnh J-type gồm có 2 trường: opcode và target.

Ví dụ: j main

Trong đó, j là ký hiệu opcode, main là địa chỉ mục tiêu.

\* Giới hạn nhảy của các câu lệnh "beq", "j"

1. Câu lệnh "beq" (Branch if Equal) có giới hạn nhảy là 16 bit, tức là nó có thể nhảy tối đa ±32768 byte so với vị trí hiện tại.
2. Câu lệnh "j" (Jump) có giới hạn nhảy là 26 bit, tức là nó có thể nhảy tới bất kỳ vị trí nào trong chương trình.

\* Độ dài mã lệnh trong kiến trúc MIPS

Độ dài mã lệnh trong kiến trúc MIPS là 32 bit. Mỗi câu lệnh MIPS được mã hóa dưới dạng một chuỗi 32 bit binary.

\* Các thành phần trong một mã lệnh

Trong MIPS, mỗi mã lệnh được gồm có nhiều thành phần chính, bao gồm:

1. Opcode: Là mã số của câu lệnh, xác định loại hoạt động cần thực hiện.
2. Rs: Là mã số của thanh ghi nguồn (source register) cho phép toán.
3. Rt: Là mã số của thanh ghi đích (destination register) cho phép toán.
4. Rd: Là mã số của thanh ghi đích (destination register) cho phép toán R-type.
5. Shamt: Là mã số cho phép dịch bit trong phép toán R-type.
6. Funct: Là mã số của phép toán trong câu lệnh R-type.
7. Offset: Là giá trị được sử dụng cho các phép toán I-type hoặc chuyển động J-type.
8. Target: Là một địa chỉ trong bộ nhớ cho biết vị trí mới của chương trình sau khi nhảy.

#### \* Khi một lệnh nhảy (branch) được thực thi do thoả mãn điều kiện nhảy/hoặc không thoả mãn điều kiện nhảy thì thanh ghi PC chứa giá trị gì?

- Khi một lệnh nhảy được thực thi, thanh ghi PC (Program Counter) chứa địa chỉ của lệnh tiếp theo mà CPU sẽ thực thi.

- Nếu không thoả mãn điều kiện nhảy, thanh ghi PC sẽ chứa giá trị của địa chỉ của câu lệnh tiếp theo trong chương trình.

#### \* - Các thành phần trong một lệnh máy, độ dài các thành phần mã lệnh loại 1 (I-Format) (VD: độ dài của phần opcode, immediate)

Mỗi lệnh máy MIPS gồm có 4 trường: opcode, rs, rt, và offset (hoặc immediate).

1. Độ dài của phần opcode: 6 bit
2. Độ dài của phần rs: 5 bit
3. Độ dài của phần rt: 5 bit
4. Độ dài của phần offset/immediate: 16 bit

Do đó, mỗi lệnh máy MIPS loại I-Format có độ dài là 32 bit.

### Chương 11: DATAPATH &CONTROL

#### \* Chức năng của các tín hiệu RegDst; RegWrite; Sign Extend; WriteData; MemWrite; MemRead; MemToReg, .......

1. RegDst (Register Destination): Xác định nơi chứa kết quả của lệnh MIPS.
2. RegWrite: Xác định xem có viết vào bộ nhớ để lưu kết quả hay không.
3. Sign Extend: Mở rộng bit dấu của số nhị phân để chuyển nó thành dạng số nguyên.
4. WriteData: Dữ liệu được viết vào bộ nhớ.
5. MemWrite: Xác định xem có viết vào bộ nhớ hay không.
6. MemRead: Xác định xem có đọc từ bộ nhớ hay không.
7. MemToReg: Xác định nơi chứa kết quả đọc từ bộ nhớ.

\* Chức năng Cờ isZero từ bộ ALU

Flag isZero (Cờ isZero) từ bộ ALU (Arithmetic Logic Unit) xác định xem kết quả của phép toán tính toán có bằng 0 hay không. Nếu kết quả bằng 0, cờ isZero được đặt là 1 (true), ngược lại nó được đặt là 0 (false). Cờ này có thể được sử dụng để điều khiển luồng xử lý của chương trình hoặc để thực hiện các phép so sánh.

\* Tín hiệu điều khiển ALUControl? (Độ dài? tạo thành từ cái gì?)

1. Tín hiệu điều khiển ALUControl là tín hiệu điều khiển cho bộ ALU (Arithmetic Logic Unit) trong máy tính. Nó điều khiển hoạt động của bộ ALU và xác định loại phép toán tính toán sẽ được thực hiện trên dữ liệu đầu vào.
2. Độ dài của tín hiệu ALUControl phụ thuộc vào số lượng phép toán tính toán mà bộ ALU có thể thực hiện và số bit dùng để mã hóa các phép toán. Thông thường, độ dài của tín hiệu ALUControl là 4 hoặc 5 bit.
3. Tín hiệu ALUControl được tạo thành từ việc kết hợp các tín hiệu điều khiển khác nhau từ hệ thống điều khiển, ví dụ như tín hiệu chỉ dẫn từ định dạng của lệnh hoặc tín hiệu chỉ dẫn từ điều khiển bảng chuyển đổi.

### Chương 12: PIPELINING

\* So sánh sự khác nhau giữa kiến trúc pipeline với kiến trúc vi xử lý thông thường?

- Trong kiến trúc pipeline, mỗi lệnh được chia thành các giai đoạn nhỏ và được xử lý một giai đoạn của nó trong mỗi chu kỳ. Điều này cho phép nhiều lệnh được xử lý đồng thời, tăng tốc độ xử lý.

- Trong kiến trúc vi xử lý thông thường, mỗi lệnh được xử lý một cách tuần tự, mỗi lệnh chờ đến khi lệnh trước nó hoàn tất mới bắt đầu được xử lý. Điều này dẫn đến tốc độ xử lý chậm hơn so với kiến trúc pipeline.

- Tóm lại, kiến trúc pipeline cho phép nhiều lệnh được xử lý đồng thời và tăng tốc độ xử lý, trong khi kiến trúc vi xử lý thông thường xử lý lệnh một cách tuần tự và tốc độ xử lý chậm hơn.

\* Hiểu kỹ về kỹ thuật pipelining

Pipelining là một kỹ thuật trong xử lý máy tính dùng để tăng hiệu suất bằng cách chia nhiệm làm việc cho các đơn vị xử lý trong máy tính và tối ưu hóa sử dụng thời gian của máy. Nó hoạt động bằng cách sắp xếp các bước trong quá trình xử lý lần lượt theo chiều dọc theo dòng của luồng dữ liệu, giúp máy tính có thể xử lý nhiều tác vụ một lúc mà không bị tắc nghẽn.

\* Hiểu các thuật ngữ data depencency, control depencency?

- Data Dependency: là mối quan hệ giữa các bước trong pipeline mà kết quả của một bước phụ thuộc vào kết quả của bước trước đó. Ví dụ, lệnh A phụ thuộc dữ liệu vào lệnh B nếu giá trị của A sử dụng giá trị đã được tính toán bởi lệnh B.

-Control dependency: là mối quan hệ giữa các lệnh trong chương trình mà thứ tự thực thi của một lệnh phụ thuộc vào kết quả của một điều kiện nhảy hoặc lệnh điều khiển khác. Ví dụ, lệnh A phụ thuộc điều khiển vào lệnh B nếu lệnh A chỉ được thực thi nếu kết quả của điều kiện nhảy hoặc lệnh điều khiển khác là "true".

\* Cách tính xung clock để thực thi một đoạn mã trên bộ vi xử lý đơn chu kỳ (single circle processor)?

1. Phân tích đoạn mã: Phân tích đoạn mã để xác định số lượng các thao tác cần được thực hiện.
2. Xác định thời gian thực hiện mỗi thao tác: Xác định thời gian mỗi thao tác cần để hoàn thành, bao gồm cả thời gian để đọc và ghi dữ liệu.
3. Tính tổng thời gian: Tổng hợp tổng thời gian cần để hoàn thành tất cả các thao tác.
4. Chia cho tần số clock: Chia tổng thời gian cho tần số clock để xác định số xung clock cần để hoàn thành đoạn mã.

Lưu ý rằng, kết quả của tính toán này chỉ là ước lượng và có thể thay đổi tùy thuộc vào số lượng dữ liệu được xử lý, tốc độ bộ nhớ và các yếu tố khác.

### Chương 13: CACHE

\* Thứ tự về tốc độ truy xuất bộ nhớ của các loại bộ nhớ (Ram, cache, ROM, HDD, SSD, Register, ....)

1. Register: Register là những vùng nhớ trong CPU, truy xuất tốc độ nhanh nhất.
2. Cache: Cache là một bộ nhớ tạm gần CPU, truy xuất tốc độ nhanh hơn bộ nhớ chính.
3. RAM: RAM là bộ nhớ chính của máy tính, truy xuất tốc độ nhanh hơn bộ nhớ cứng nhưng chậm hơn cache. SSD:
4. SSD là một loại bộ nhớ cứng sử dụng những đầu đọc/ghi flash, truy xuất tốc độ nhanh hơn HDD.
5. HDD: HDD là một loại bộ nhớ cứng sử dụng đĩa cứng, truy xuất tốc độ chậm hơn SSD.
6. ROM: ROM là bộ nhớ chứa dữ liệu không thay đổi, truy xuất tốc độ chậm hơn RAM.

Lưu ý: Thứ tự tốc độ truy xuất này có thể khác nhau tùy theo kiểu máy tính và cấu hình.

#### \* Chức năng Bộ nhớ Cache level 1, level 2

1. Cache Level 1 (L1 Cache): là bộ nhớ Cache gần nhất với CPU và có tốc độ truy xuất nhanh nhất trong tất cả các Cache level. Nó được sử dụng để lưu trữ dữ liệu và lệnh mà CPU sẽ sử dụng nhiều nhất.
2. Cache Level 2 (L2 Cache): là bộ nhớ Cache tiếp theo với tốc độ truy xuất chậm hơn so với L1 Cache, nhưng nhanh hơn so với RAM. Nó được sử dụng để lưu trữ dữ liệu và lệnh mà CPU có thể sử dụng trong tương lai.

\* Spatial locality, temporal locality là gì?

Spatial locality và Temporal locality là hai khái niệm quan trọng trong lĩnh vực bộ nhớ Cache và tối ưu hóa hiệu suất bộ vi xử lý.

1. Spatial locality (Sẵn có vị trí): là một khái niệm mô tả việc truy xuất các địa chỉ bộ nhớ gần nhau trong một khoảng thời gian ngắn. Spatial Locality cho biết rằng khi một địa chỉ bộ nhớ được truy xuất, các địa chỉ bộ nhớ gần nó cũng có khả năng được truy xuất trong tương lai.
2. Temporal locality (Sẵn có thời gian): là một khái niệm mô tả việc truy xuất cùng một địa chỉ bộ nhớ trong một khoảng thời gian ngắn. Temporal Locality cho biết rằng khi một địa chỉ bộ nhớ được truy xuất, nó có khả năng được truy xuất lại trong tương lai.

\*Conflict miss, cold miss, miss penalty được sử dụng trong trường hợp nào?

- Conflict miss (miss xung đột): Đây là trường hợp khi hai hoặc nhiều địa chỉ bộ nhớ cùng muốn lưu trữ trong cùng một vùng bộ nhớ Cache có kích thước hạn chế. Do đó, một trong số các địa chỉ này phải bị loại bỏ để tạo ra chỗ trống cho địa chỉ khác.

- Cold miss (miss lạnh): Đây là trường hợp khi một địa chỉ bộ nhớ được truy xuất lần đầu tiên và không tìm thấy trong bộ nhớ Cache. Trong trường hợp này, bộ nhớ Cache phải truy xuất bộ nhớ RAM để lấy dữ liệu và lưu trữ trong bộ nhớ Cache.

- Miss penalty (phạt miss): Đây là thời gian mà bộ nhớ Cache phải chờ đợi khi truy xuất bộ nhớ RAM để lấy dữ liệu khi xảy ra miss. Thời gian này càng lâu càng tốn tài nguyên và giảm hiệu suất của hệ thống.

#### \* Cấu tạo của từng bộ nhớ? (VD: Bộ nhớ Cache được cấu tạo từ loại phần tử nhớ nào: SDRAM hay DRAM hay SSD hay HDD, ....)

1. Bộ nhớ Cache: Bộ nhớ Cache thường được cấu tạo từ SRAM (Static Random Access Memory), một loại bộ nhớ không bị mất dữ liệu khi không có nguồn điện. Bộ nhớ SRAM được sử dụng vì nó có tốc độ truy xuất nhanh hơn và yêu cầu nguồn điện ít hơn so với DRAM (Dynamic Random Access Memory), loại bộ nhớ được sử dụng cho bộ nhớ RAM chính.
2. Bộ nhớ RAM: Bộ nhớ RAM thường được cấu tạo từ DRAM hoặc SDRAM (Synchronous Dynamic Random Access Memory), một loại bộ nhớ đồng bộ với tốc độ clock của hệ thống.
3. Bộ nhớ ROM: Bộ nhớ ROM (Read-Only Memory) thường được cấu tạo từ PROM (Programmable Read-Only Memory), EPROM (Erasable Programmable Read-Only Memory) hoặc EEPROM (Electrically Erasable Programmable Read-Only Memory). Chúng ta không thể ghi dữ liệu mới vào bộ nhớ ROM, nhưng chúng ta có thể sửa đổi nội dung trong bộ nhớ EPROM hoặc EEPROM bằng cách sử dụng một thiết bị ghi.
4. HDD (Hard Disk Drive): Các ổ đĩa cứng (HDD) được cấu tạo từ các đĩa mạ vàng và đầu đọc ghi, và sử dụng từng đầu đọc ghi để quét qua đĩa mạ vàng để truy xuất dữ liệu.
5. SSD (Solid State Drive): Các ổ đĩa cứng quảng cáo (SSD) được cấu tạo từ các tấm NAND Flash Memory và đầu đọc ghi, và không cần đĩa mạ vàng hoặc đầu đọc ghi di động để truy xuất dữ liệu.

#### \* Các kỹ thuật ánh xạ bộ nhớ Cache (Set Associate Mapping, 4-ways Set Associate Mapping, Direct Mapping, ...)

1. Direct Mapping: là kỹ thuật ánh xạ cache đơn giản nhất. Mỗi block trong bộ nhớ chính được ánh xạ với một block trong cache.
2. Set Associate Mapping: là một trong những kỹ thuật ánh xạ cache phổ biến nhất. Trong kỹ thuật này, các block trong bộ nhớ chính được chia thành các tập và mỗi tập được ánh xạ với một hoặc nhiều block trong cache.
3. 4-ways Set Associate Mapping: là một dạng mở rộng của Set Associative Mapping, trong đó mỗi tập được ánh xạ với 4 block trong cache. Kỹ thuật này cung cấp tốc độ truy xuất tốt hơn so với Set Associative Mapping.

#### \* Cách tính thời gian truy xuất trung bình của bộ vi xử lý? (VD: Một bộ vi xử lý có tỉ lệ trật cache (cache miss) là 65%. Thời gian truy xuất bộ nhớ cache là 2 ns, thời gian truy xuất bộ nhớ chính là 18 ns. Thời gian truy xuất trung bình của bộ vi xử lý này là bao nhiêu?)

Thời gian truy xuất trung bình của bộ vi xử lý = (Tỷ lệ cache miss) \* (Thời gian truy xuất bộ nhớ chính) + (1 - Tỷ lệ cache miss) \* (Thời gian truy xuất bộ nhớ cache)

Ví dụ:

Tỷ lệ cache miss = 65%

Thời gian truy xuất bộ nhớ cache = 2 ns

Thời gian truy xuất bộ nhớ chính = 18 ns

Thời gian truy xuất trung bình = (65%) \* (18 ns) + (35%) \* (2 ns) = 11.7 ns.

### Chương 14: VIRTUAL MEMORY & INPUT/OUTPUT

\* Định nghĩa về Bộ nhớ ảo; các thành phần nào quản lý bộ nhớ ảo? Ưu điểm của bộ nhớ ảo?

- Bộ nhớ ảo (Virtual Memory) là một loại bộ nhớ giả mạo, tạo ra một không gian bộ nhớ lớn hơn cho chương trình đang chạy, cho phép chương trình truy cập vào nhiều dữ liệu hơn so với bộ nhớ RAM thực tế. Bộ nhớ ảo được quản lý bởi hệ điều hành (OS) và bộ điều chỉnh (MMU) trong vi xử lý.

- Ưu điểm của bộ nhớ ảo:

*+ Cho phép chương trình truy cập vào nhiều dữ liệu hơn so với bộ nhớ RAM thực tế.*

*+ Giúp giảm tải trên bộ nhớ RAM và cho phép chương trình chạy mượt mà hơn.*

*+ Giúp tránh việc tràn bộ nhớ khi chương trình yêu cầu nhiều dữ liệu hơn so với số lượng bộ nhớ RAM có sẵn.*

- Các thành phần quản lý bộ nhớ ảo bao gồm:

+ Hệ điều hành (Operating System): quản lý việc chuyển dữ liệu giữa bộ nhớ RAM và bộ nhớ cứng.

+ Memory Management Unit (MMU): thiết bị được gắn trên bộ vi xử lý và chuyển đổi địa chỉ trong bộ nhớ ảo thành địa chỉ trong bộ nhớ RAM hoặc bộ nhớ cứng.

\* Page fault, page table, Thuật toán thay trang LRU (Least Recently Used)/ FIFO là gì?

- Page Fault: là một tình trạng xảy ra khi hệ thống bộ vi xử lý muốn truy cập một trang nhớ được yêu cầu nhưng không tìm thấy trong bộ nhớ. Điều này cần thiết phải đưa trang đó từ bộ nhớ trung gian vào bộ nhớ.

- Page Table: Là một bảng cố định cho phép hệ thống tìm kiếm và xác định vị trí trang nhớ của một trang cụ thể trong bộ nhớ.

- Thuật toán thay trang LRU (Least Recently Used): là một thuật toán cho phép quản lý trang trong bộ nhớ và xác định trang nào sẽ được thay thế trong trường hợp quá tải. LRU sẽ xác định trang đã được sử dụng ít nhất gần đây và thay thế nó bằng trang mới.

- Thuật toán thay trang FIFO (First In First Out): là một thuật toán cho phép quản lý trang trong bộ nhớ và xác định trang nào sẽ được thay thế trong trường hợp quá tải. FIFO sẽ xác định trang được tải vào trước nhất và thay thế nó bằng trang mới.

#### \* Liệt kê các thiết bị xuất/nhập (IO).

1. Màn hình (Monitor)
2. Bàn phím (Keyboard)
3. Chuột (Mouse)
4. Cảm biến (Sensor)
5. Thiết bị nhập/xuất âm thanh (Audio input/output devices)
6. Thiết bị nhập/xuất hình ảnh (Image input/output devices)
7. Thiết bị ghi đĩa (Disc drive)
8. USB (Universal Serial Bus)
9. Ethernet
10. Wi-Fi
11. Bluetooth
12. Các thiết bị đọc thẻ nhớ (Memory card reader)
13. Các thiết bị in (Printer)
14. Joystick
15. Gamepad.

\* Chức năng, nhiệm vụ của BUS trong hệ thống máy tính? tuyến BUS nào có tốc độ Cao nhất/Thấp nhất?

- BUS trong hệ thống máy tính là một tuyến kết nối giữa các thành phần trong máy tính.

- Chức năng chính của BUS là truyền dữ liệu giữa các thành phần, điều khiển hoạt động của các thiết bị và cung cấp nguồn điện cho các thành phần. BUS giúp giảm thiểu số lượng dây kết nối, giảm thiểu sự phức tạp và tăng tốc độ truyền dữ liệu trong hệ thống máy tính.

Tuyến BUS nào có tốc độ cao nhất/thấp nhất:

- Tuyến BUS Front Side Bus (FSB) có tốc độ cao nhất trong các tuyến BUS trong máy tính.

- Tuyến BUS PCI (Peripheral Component Interconnect) có tốc độ thấp hơn so với FSB.