**Tổng hợp kiến thức**

**1. Cấu trúc của vi xử lý**

**1.1. Cấu trúc cơ bản của 1 bộ vi xử lý**

* **Đơn vị xử lý trung tâm CPU** (Central Processing Unit):

CPU có cấu tạo gồm có đơn vị xử lý số học và lôgic (ALU), các thanh ghi, các khối lôgic và các mạch giao tiếp. Chức năng của CPU là tiến hành các thao tác tính toán xử lý, đưa ra các tín hiệu địa chỉ, dữ liệu và điều khiển nhằm thực hiện một nhiệm vụ nào đó do người lập trình đưa ra thông qua các lệnh (Instructions).

* **Các bộ nhớ** (Memories):

Có hai loại bộ nhớ là bộ nhớ chương trình và bộ nhớ dữ liệu. Bộ nhớ chương trình dùng để chứa mã chương trình hướng dẫn CPU thực hiện một nhiệm vụ nào đó. Thông thường thì bộ nhớ chương trình là các loại bộ nhớ “không bay hơi” (non-volatile), nghĩa là không bị mất nội dung chứa bên trong khi ngừng cung cấp nguồn nuôi. Có thể kể ra một số bộ nhớ thuộc loại này như: ROM, PROM, EPROM, EEPROM, Flash. Bộ nhớ dữ liệu là bộ nhớ dùng để chứa dữ liệu (bao gồm các tham số, các biến tạm thời…). Tuỳ thuộc loại dữ liệu mà bộ nhớ loại này có thể là loại “không bay hơi” hoặc “bay hơi” (mất dữ liệu khi cắt nguồn nuôi). Loại “bay hơi” thường thấy là các bộ nhớ SRAM.

* **Các cổng vào/ra song song** (Parallel I/O Ports):

Đây là các đường tín hiệu được nối với một số chân của IC dùng để giao tiếp với thế giới bên ngoài IC. Giao tiếp ở đây là đưa điện áp ra hoặc đọc vào giá trị điện áp tại chân cổng. Các giá trị điện áp đưa ra hay đọc vào chỉ có thể đwocj biểu diễn bởi một trong hai giá trị lôgic (0 hoặc 1). Trong kỹ thuật vi xử lý, người ta thường dùng quy ước lôgic dương: giá trị lôgic 0 ứng với mức điện áp thấp xấp xỉ 0VDC, giá trị lôgic 1 ứng với mức điện áp cao xấp xỉ +5VDC. Tùy loại vi xử lý mà “khoảng xấp xỉ” đó là khác nhau nhưng nhìn chung là tương thích với mức lôgic TTL. Mỗi cổng vào/ra song song thường gồm 08 đường vào/ra khác nhau và gọi là các cổng 08 bit. Các đường tín hiệu vào/ra của các cổng và thuộc cùng một cổng là độc lập với nhau. Điều đó có nghĩa là ta có thể đưa ra hay đọc vào các giá trị lôgic khác nhau đối với từng chân cổng (từng đường tín hiệu vào/ra). Một điều cần chú ý nữa đối với các cổng vào/ra đó là chúng có thể được tích hợp thêm (nói đúng hơn là kiêm thêm) các chức năng đặc biệt liên quan đến các ngoại vi khác.

* **Các cổng vào/ra nối tiếp** (Serial I/O Ports):

Khác với cổng song song, với cổng nối tiếp các bit dữ liệu được truyền lần lượt trên cùng một đường tín hiệu thay vì truyền cùng một lúc trên các đường tín hiệu khác nhau. Thông thường thì việc truyền dữ liệu bằng cổng nối tiếp phải tuân theo một cơ chế, một giao thức hay một nguyên tắc nhất định. Có thể kể ra một số giao thức như SPI, I2C, SCI…

**Cổng nối tiếp có 02 kiểu truyền dữ liệu chính:**

**- Truyền đồng bộ (synchronous):** thiết bị truyền và thiết bị nhận đều dùng chung một xung nhịp (clock).

**- Truyền dị bộ (asynchronous):**thiết bị truyền và thiết bị nhận sử dụng hai nguồn xung nhịp riêng. Tuy nhiên hai nguồn xung nhịp này không được khác nhau quá nhiều.

Xung nhịp là yếu tố không thể thiếu trong truyền dữ liệu nối tiếp và nó có vaitrò xác định giá trị của bit dữ liệu (hay nói đúng hơn là xác định thời điểm đọc mức lôgic trên đường truyền dữ liệu).

**Cổng nối tiếp có thể có một trong các tính năng sau:**

***-Đơn công:*** thiết bị chỉ có thể hoặc truyền hoặc nhận dữ liệu.

***-Bán song công:*** thiết bị có thể truyền và nhận dữ liệu nhưng tại một thời điểm chỉ có thể làm một trong hai việc đó.

**-Song công:** thiết bị có thể truyền và nhận dữ liệu đồng thời.

* **Các bộ đếm/bộ định thời** (Timers):

Là các ngoại vi được thiết kế để thực hiện một nhiệm vụ đơn giản: đếm các xung nhịp. Mỗi khi có thêm một xung nhịp tại đầu vào đếm thì giá trị của bộ đếm sẽ được tăng lên 01 đơn vị (trong chế độ đếm tiến/đếm lên) hay giảm đi 01 đơn vị (trong chế độ đếm lùi/đếm xuống).

Xung nhịp đưa vào đếm có thể là một trong hai loại: Xung nhịp bên trong IC. Đó là xung nhịp được tạo ra nhờ kết hợp mạch dao động bên trong IC và các linh kiện phụ bên ngoài nối với IC. Ta có thể ví đó là “nhịp tim” để toàn bộ các phần cứng bên trong vi xử lý (bao gồm cả CPU và các ngoại vi) có thể hoạt động được. Trong trường hợp sử dụng xung nhịp loại này, người ta gọi là các bộ định thời (timers). Do xung nhịp bên loại này thường đều đặn nên ta có thể dùng để đếm thời gian một cách khá chính xác. Xung nhịp bên ngoài IC. Đó là các tín hiệu lôgic thay đổi liên tục giữa 02 mức 0-1 và không nhất thiết phải là đều đặn. Trong trường hợp này người ta gọi là các bộ đếm (counters). Ứng dụng phổ biến của các bộ đếm là đếm các sự kiện bên ngoài như đếm các sản phầm chạy trên băng chuyền, đếm xe ra/vào kho bãi…

**1.2. Các thanh ghi**

Số lượng, kích cỡ và kiểu của các thanh ghi thay đổi từ vi xử lý này sang vi xử lý khác. Tuy nhiên, các thanh ghi này thực hiện các thao tác tương tự nhau. Cấu trúc các thanh ghi đóng vai trò quan trọng trong việc thiết kế kiến trúc của vi xử lý. Đồng thời, cấu trúc thanh ghi với một loại vi xử lý cụ thể cho biết mức độ thuận lợi và dễ dùng khi lập trình cho vi xử lý đó. Dưới đây là các thanh ghi cơ bản nhất:

* Thanh ghi lệnh: lưu mã lệnh đang thực hiện. Sau khi nạp mã lệnh từ bộ nhớ, vi xử lý lưu mã lệnh trong thanh ghi lệnh. Giá trị trong thanh ghi này luôn được vi xử lý giải mã để xác định lệnh. Kích cỡ từ (word) của vi xử lý quyết định kích cỡ của thanh ghi này. Ví dụ, vi xử lý 32 bít thì sẽ có thanh ghi lệnh 32 bít.
* Bộ đếm chương trình: chứa địa chỉ của lệnh hay mã thực thi (op-code) tiếp theo được thực hiện. Thanh ghi này có đặc điểm sau:
* Khi một chương trình được nạp vào bộ nhớ, địa chỉ ô nhớ chứa lệnh đầu tiên của chương trình được nạp vào thanh ghi này.
* Để thực hiện lệnh, vi xử lý nạp nội dung của bộ đếm chương trình vào bus địa chỉ và đọc ô nhớ ở địa chỉ đó. Giá trị của bộ đếm chương trình tự động tăng theo bộ lô-gíc trong của vi xử lý. Như vậy, vi xử lý thực hiện các lệnh tuần tự trừ phi chương trình có các lệnh làm thay đổi trật tự thực hiện.
* Kích cỡ của bộ đếm chương trình phụ thuộc vào kích cỡ của bus địa chỉ.
* Có một số lệnh (như các lệnh nhảy) làm thay đổi nội dung của bộ đếm chương trình so với trình tự thông thường. Khi đó, giá trị của thanh ghi được xác định thông qua địa chỉ được chỉ định trong các lệnh này.
* Thanh ghi địa chỉ bộ nhớ: chứa địa chỉ của dữ liệu. Vi xử lý sử dụng địa chỉ này như là các con trỏ trực tiếp tới bộ nhớ. Nội dung của ô nhớ có địa chỉ này chính là dữ liệu đang được trao đổi và xử lý.
* Thanh ghi tổng: còn được gọi là thanh ghi tích lũy (accumulator). Thanh ghi này có kích thước 8, 16, 32 hoặc 64 bít và thường dùng thể lưu kết quả tính toán của đơn vị xử lý số học và lô-gíc ALU. Thanh ghi này còn được dùng để trao đổi dữ liệu với các thiết bị vào/ra.

**2. Bộ xử lý Intel 8086**

**2.1. Các thanh ghi 8086**

**a. Các thanh ghi đoạn**

* Thanh ghi đoạn mã CS (Code-Segment).
* Thanh ghi đoạn dữ liệu DS (Data Segment).
* Thanh ghi đoạn ngăn xếp SS (Stack Segment).
* Thanh ghi đoạn dữ liệu phụ ES (Extra Segment).

**b. Các thanh ghi đa năng**

* AX (Accumulator): thanh ghi tích lũy. Các kết quả của các thao tác thường được chứa ở AX (kết quả của phép nhân, chia).
* BX (Base): thanh ghi cơ sở, thường dùng để chứa địa chỉ cơ sở của một dãy các ô nhớ.
* CX (Count): thanh đếm. CX thường được dùng để chứa số lần lặp trong trường hợp các lệnh LOOP (lặp).
* DX (Data): thanh ghi dữ liệu. DX tham gia các thao tác của phép nhân hoặc chia các số 16 bít. DX thường dùng để chứa địa chỉ của các cổng trong các lệnh vào/ ra dữ liệu.

**c. Các thanh ghi con trỏ và chỉ số**

* IP: con trỏ lệnh (Instruction Pointer). IP luôn trỏ vào lệnh tiếp theo sẽ được thực hiện nằm trong đoạn mã CS. Địa chỉ đầy đủ của lệnh tiếp theo có dạng CS:IP và được xác định như trình bày trong phần 1.2.
* BP: con trỏ cơ sở (Base Pointer). BP luôn trỏ vào một dữ liệu nằm trong đoạn ngăn xếp SS. Địa chỉ đầy đủ của một phần tử trong đoạn ngăn xếp có dạng SS:BP và được xác định như trình bày trong phần 1.2.
* SP: con trỏ ngăn xếp (Stack Pointer). SP luôn trỏ vào đỉnh hiện thời của ngăn xếp nằm trong đoạn ngăn xếp SS. Địa chỉ đỉnh ngăn xếp có dạng SS:SP và được xác định như trình bày trong phần 1.2.
* SI: chỉ số gốc hay nguồn (Source Index). SI chỉ vào dữ liệu trong đoạn dữ liệu DS mà địa chỉ cụ thể đầy đủ có dạng DS:SI và được xác định như trình bày trong phần 1.2.
* DI: chỉ số đích (Destination Index). DI chỉ vào dữ liệu trong đoạn dữ liệu DS (hoặc ES) mà địa chỉ cụ thể đầy đủ có dạng DS:DI (hoặc ES:DI)

d. Thanh ghi cờ FR (flag register)

* U là các bít không sử dụng.
* C hoặc CF (Carry Flag): cờ nhớ. CF = 1 khi có nhớ hoặc mượn từ bít có nghĩa lớn nhất MSB (Most Significant Bit).
* P hoặc PF (Parity Flag): cờ chẵn lẻ. PF phản ánh tính chẵn lẻ của tổng số bít 1 có trong kết quả. Cờ PF =1 khi tổng số bít 1 trong kết quả là lẻ (odd parity) và PF =0 khi tổng số bít 1 trong kết quả là chẵn (even parity).
* A hoặc AF (Auxiliary Carry Flag): cờ nhớ phụ rất có ý nghĩa khi ta làm việc với các số BCD (Binary Coded Decimal). AF = 1 khi có nhớ hoặc mượn từ một số BCD thấp (4 bít thấp) sang một số BCD cao (4 bít cao).
* Z hoặc ZF (Zero Flag): cờ rỗng. ZF =1 khi kết quả = 0 và ZF =0 khi kết quả # 0.
* S hoặc SF (sign flag): cờ dấu. SF = 1 khi kết quả âm và SF = 0 khi kết quả không âm.
* hoặc OF (Overflow Flag): cò tràn. OF = 1 khi kết quả là một số bù 2 vượt qua ngoài giới hạn biểu diễn dành cho nó.

Ngoài ra, bộ vi xử lý 8086 còn có các cờ điều khiển sau đây:

* T hoặc TF (Trap Flag): cờ bẫy. TF = 1 thì CPU làm việc ở chế độ chạy từng lệnh (chế độ này dùng khi cần tìm lỗi trong một chương trình).
* I hoặc IF (Interrupt Enable Flag): cờ cho phép ngắt. IF = 1 thì CPU cho phép các yêu cầu ngắt (che được) và IF = 0 thì CPU cấm các yêu cầu ngắt.
* D hoặc DF (Direction Flag): cờ hướng. DF = 1 khi CPU làm việc với chuỗi ký tự theo thứ tự từ phải sang trái, hoặc giảm địa chỉ (vì vậy D chính là cờ lùi) và DF = 0 khi CPU làm việc với chuỗi ký tự theo thứ tự từ trái sang phải, hoặc tăng địa chỉ .

**2.2. Phân đoạn bộ nhớ của 8086**

Vi xử lí chia không gian 1Mbyte bộ nhớ thành các vùng khác nhau theo nội dung mà chúng lưu trữ, gồm các vùng nhớ để:

* Chứa mã chương trình.
* Chứa dữ liệu và kết quả của chương trình.
* Tạo một vùng nhớ đặc biệt gọi là ngăn xếp (stack) dùng vào việc quản lý các thông số của bộ vi xử lý khi gọi thự hiện các chương trình con hoặc trở về từ chương trình con.

**Địa chỉ vật lý = Thanh\_ghi\_đọan × 10H (16 hệ 10) + Thanh\_ghi\_lệch**

**Thanh\_ghi\_đoạn: Thanh\_ghi\_lệch hay segment: offset**

**2.3. Các chế độ địa chỉ:**

* **Chế độ địa chỉ thanh ghi**

Đặc điểm: các toán hạng đều là các thanh ghi

MOV đích, gốc. (chuyển dữ liệu từ đích vào gốc)

* **Chế độ địa chỉ tức thì**

Đặc điểm: toán hạng gốc là 1 hằng, toán hạng còn lại có thể là thanh ghi hoặc ô nhớ

* **Chế độ địa chỉ trực tiếp**

Đặc điểm sử dụng 1 hằng để biểu diễn địa chỉ 1 ô nhớ là 1 toán hạng. Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 hằng (không được là 1 ô nhớ)

* **Chế độ địa chỉ gián tiếp qua thanh ghi**

Đặc điểm: sử dụng 1 thanh ghi(BX, BP, SI, DI) để chứa địa chỉ của 1 ô nhớ là 1 toán hạng. Toán hạng còn lại có thể là 1 thanh ghi hoặc 1 hằng (không được là 1 ô nhớ)

* **Chế độ địa chỉ tương đối cơ sở**

Đặc điểm: 1 toán hạng là 1 địa chỉ ô nhớ, được biểu diễn bởi 2 thành phần: 1 thanh ghi cơ ở(BX,BP) và 1 hằng. Toán hạng còn lại là 1 thanh ghi hoặc hằng (không được là 1 ô nhớ)

Nếu thanh ghi là BX thì ô nhớ thuộc đoạn DS

Nếu thanh ghi là BP thì ô nhớ thuộc đoạn SS

* **Chế độ địa chỉ chỉ số**

Đặc điểm: 1 toán hạng là 1 địa chỉ ô nhớ gồm 2 thành phần: thanh ghi chỉ số (SI,DI) và hằng. Toán hạng còn lại là thanh ghi hoặc hằng (không được là 1 ô nhớ)

* **Chế độ địa chỉ cơ sở**

Đặc điểm: 1 toán hạng là địa chỉ ô nhớ gồm 3 phần: thanh ghi cơ sở (BX,BP), thanh ghi chỉ số (SI,DI), và hằng. Toán hạng còn lại là thanh ghi hoặc hằng (không được là 1 ô nhớ)

**2.4. Tập lệnh Intel 8086**

**2.4.1. Các lệnh trao đổi dữ liệu**

a. MOV – Chuyển 1 byte hay word

Viết lệnh: MOV Đích, Gốc.

Mô tả: Đích <= Gốc

b. LEA - Nạp địa chỉ hiệu dụng vào thanh ghi

Viết lệnh: LEA Đích, Gốc

Trong đó:

+ Đích thường là một trong các thanh ghi: BX, CX, DX, BP, SI, DI.

+ Gốc là tên biến trong đoạn DS được chỉ rõ trong lệnh hoặc ô nhớ cụ thể.

Mô tả:

Đích <= Địa chỉ lệch của Gốc, hoặc

Đích <= Địa chỉ hiệu dụng của Gốc

c. IN- Đọc dữ liệu từ cổng vào thanh ghi ACC.

Viết lệnh: IN ACC, Port

Mô tả: ACC <= [Port]

d. OUT - Ghi dữ liệu từ Acc ra cổng

Viết lệnh: OUT Port, Acc

Mô tả: Acc => [Port]

**2.4.2. Các lệnh tính toán số học và logic**

a. ADD-Cộng 2 toán hạng

Viết lệnh: ADD Đích, Gốc.

Mô tả: Đích <= Đích + Gốc.

b. SUB-Trừ 2 toán hạng

Viết lệnh: SUB Đích, Gốc.

Mô tả: Đích <= Đích - Gốc.

c. MUL - Nhân số không dấu

Viết lệnh: MUL Gốc

Trong đó toán hạng Gốc là số nhân và có thể tìm được theo các chế độ địa chỉ khác nhau.

Mô tả: tuỳ theo độ dài của toán hạng Gốc ta có 2 trường hợp tổ chức phép nhân, toán hạng số nhân và nơi chứa kết quả:

d. DIV – Chia nguyên 2 số không có dấu

Viết lệnh: DIV Gốc

Trong đó toán hạng Gốc là số chia và có thể tìm được theo các chế độ địa chỉ khác nhau.

Mô tả: tuỳ theo độ dài của toán hạng gốc ta có 2 trường hợp bố trí phép chia, toán hạng số bị chia và nơi lưu kết quả:

e. CMP- So sánh 2 byte hay 2 word

Viết lệnh: CMP Đích, Gốc.

Mô tả: Đích – Gốc.

f. AND - Phép và 2 toán hạng

Viết lệnh: AND Đích, Gốc

Mô tả: Đích - Đích, Gốc.

**2.4.3. Điều khiển, rẽ nhánh và lặp**

Các lệnh rẽ nhánh và lặp tiêu biểu



a. JMP - Nhảy (vô điều kiện) đến một đích nào đó

b. LOOP -Lặp lại đoạn chương trình do nhãn chỉ ra cho đến khi CX=0

Viết lệnh: LOOP NHAN

c. Các lệnh nhảy có điều kiện với dữ liệu có dấu:

Lệnh **JG:**             Nhảy nếu [Đích] > [Nguồn] ; (SF = 0F và ZF = 0)

Lệnh **JL:**             Nhảy nếu [Đích] < [Nguồn] ; (SF <> 0F)

Lệnh **JGE:**Nhảy nếu [Đích] ≥[Nguồn] ; (SF = 0F)

Lệnh **JLE:**Nhảy nếu [Đích] ≤[Nguồn] ; (CF <> 0F và ZF = 1)

Trong đó: [Đích] và [Nguồn] chính là hai toán hạng: [Toán hạng đích] và [Toán hạng nguồn] trong lệnh Cmp đứng ngay trước lệnh nhảy. Tức là, chương trình sử dụng lệnh Cmp để tạo điều kiện nhảy cho các lệnh này. Cụ thể: lệnh nhảy có thực hiện được hay không (có chuyển điều khiển đến <Vị trí đích> hay không) phụ thuộc vào giá trị của [Đích] và [Nguồn] trong lệnh Cmp đứng ngay trước nó.

d. **Các lệnh nhảy có điều kiện với dữ liệu không dấu:**

Lệnh **JA:**            Nhảy nếu [Đích] > [Nguồn] ; (CF = 0 và ZF = 0)

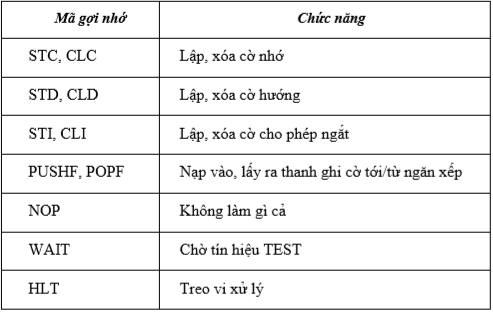
Lệnh **JB:**Nhảy nếu [Đích] < [Nguồn] ; (CF = 0)

Lệnh **JNA:**Nhảy nếu [Đích] không lớn hơn [Nguồn]; (CF =1 or ZF =1)

Lệnh **JNB:**Nhảy nếu [Đích] không nhỏ hơn [Nguồn] ; (CF = 0)

**2.4.4. Điều khiển vi xử lý**

Các lệnh điều khiển vi xử lý tiêu biểu



**2.5. Ngắt và xử lý ngắt trong 8086**

**2.5.1. Ngắt, sự cần thiết của ngắt trong CPU**

Ngắt là việc tạm dừng việc chương trình đang chạy để CPU có thể chạy một chương trình khác nhằm xử lý một yêu cầu do bên ngoài đưa tới CPU như yêu cầu vào/ra hoặc do chính yêu cầu của bên trong CPU như lỗi trong khi tính toán. Trong cách tổ chức trao đổi dữ liệu thông qua việc thăm dò trạng thái sẵn sàng của thiết bị ngoại vi, trước khi tiến hành bất kỳ một cuộc trao đổi dữ liệu nào CPU phải dành toàn bộ thời gian vào việc xác định trạng thái sẵn sàng làm việc của thiết bị ngoại vi. Để tận dụng khả năng của CPU để làm thêm được nhiều công việc khác nữa, chỉ khi nào có yêu cầu trao đổi dữ liệu thì mới yêu cầu CPU tạm dừng công việc hiện tại để phục vụ việc trao đổi dữ liệu. Sau khi hoàn thành việc trao đổi dữliệu thì CPU lại phải quay về để làm tiếp công việc hiện đang bị gián đoạn.

**2.5.2. Các loại ngắt 8086**

Trong hệ vi xử lý 8086 có thể xếp các nguyên nhân gây ra ngắt CPU vào 3 nhóm như

sau:

* Nhóm các ngắt cứng: đó là các yêu cầu ngắt CPU do các tín hiệu đến từ các chân INTR và NMI. Ngắt cứng INTR là yêu cầu ngắt che được. Các lệnh CLI và STI có ảnh hưởng trực tiếp tới trạng thái của cờ IF trong bộ vi xử lý, tức là ảnh hưởng tới việc CPU có nhận biết yêu cầu ngắt tại chân tín hiệu này hay không. Yêu cầu ngắt tại chân INTR có thể có kiểu ngắt N nằm trong khoảng 0-FFH. Kiểu ngắt này phải được đưa vào bus dữ liệu để CPU có thể đọc được khi có xung trong chu kỳ trả lời chấp nhận ngắt. Các ngắt NMI là các ngắt không che được, tức là không chịu ảnh hưởng của cờ ngắt.
* Nhóm các ngắt mềm: khi CPU thực hiện các lệnh ngắt dạng INT N, trong đó N là số hiệu (kiểu) ngắt nằm trong khoảng 00-FFH (0-255).
* Nhóm các hiện tượng ngoại lệ: đó là các ngắt do các lỗi nảy sinh trong quá trình hoạt động của CPU như phép chia cho 0, xảy ra tràn khi tính toán.

**2.5.3. Xử lý ưu tiên khi ngắt**

Mức ưu tiên ngăt như sau:

+ ngắt trong: INT 0 (phép chia cho 0), INT N,.. . . . cao nhất

+ ngắt không che được NMI

+ ngắt che được INTR

+ ngắt để chạy từng lệng INT 1 . . . thấp nhất

Theo thứ tự ưu tiên ngầm định trong việc xử lý ngắt của CPU 8086 thì INT 0 có mức ưu tiên cao hơn INTR, vì vậy đầu tiên CPU sẽ thực hiện chương trình phục vụ ngắt INT 0 để đáp ứng với lỗi đặc biệt cho phép chia cho 0 gây ra và cờ IF bị xóa về 0. Yêu cầu ngắt INTR sẽ tự động bị cấm cho tới khi chương trình phục vụ ngắt INT 0 được hoàn tất và trở về nhờ IRET, cờ IF cũ được trả lại. Tiếp theo đó CPU sẽ đáp ứng yêu cầu ngắt INTR bằng cách thực hiện chương trình phục vụ ngắt dành cho INTR.

**3. Lập trình hợp ngữ 8086**

**3.1. Khung chương trình, cú pháp, cách tạo và chạy chương trình hợp ngữ**

**a. Khung chương trinh**

**TITLE**  Chương trình hợp ngữ

**.MODEL** Kiểu kích thước bộ nhớ ; Khai báo quy mô sử d ụng bộ nhớ

**.STACK** Kích thước ; Khai báo dung lượng đoạn stack

**.DATA** ; Khai báo đoạn dữ liệu msg DB 'Hello$'

**.CODE** ; Khai báo đoạn mã

**main PROC**

…

**CALL** Subname ; Gọi chương trình con

…

**main ENDP**

**b. Cú pháp**

Một dòng lệnh trong chương trình hợp ngữ gồm có các trường (field) sau (không nhất thiết phải đầy đủ tất cả các trường):

Tên Lệnh Toán hạng Chú thích

A: MOV AH,10h ; Đưa giá trị 10h vào thanh ghi AH

Main PROC

Trường tên chứa nhãn, tên biến hay tên thủ tục. Các tên nhãn có thể chứa tối đa 31 ký tự, không chứa ký tự trắng (space) và không được bắt đầu bằng số (A: hay Main:). Các nhãn được kết thúc bằng dấu ':'.

Trường lệnh chứa các lệnh sẽ thực hiện. Các lệnh này có thể là các lệnh thật (MOV) hay các lệnh giả (PROC). Các lệnh thật sẽ được dịch ra mã máy. Trường toán hạng chứa các toán hạng cần thiết cho lệnh (AH,10h).

Trường chú thích phải được bắt đầu bằng dấu ';'.

Trường này chỉ dùng cho người lập trình để ghi các lời giải thích cho chương trình. Chương trình dịch sẽ bỏ qua các lệnh nằm phía sau dấu ;

**c. Cách tạo và chạy 1 chương trình hợp ngữ**

Như đã nói trong phần trước, máy IBM PC là phương tiện lý tưởng để chúng ta tạo ra và thử nghiệm các chương trình hợp ngữ 8086. Các bước bao gồm:

1. Dùng các phần mềm soạn thảo văn bản để tạo ra một tệp văn bản chương trình gốc bằng hợp ngữ. Tệp này phải được gán đuôi .ASM.

2. Dùng chương trình dịch MASM để dịch tệp .ASM ra mã máy dưới dạng tệp .OBJ. Nếu trong bước này nếu trong chương trình có lỗi cú pháp thì phải quay lại bước 1 để sửa lại chương trình gốc.

3. Dùng chương trình LINK để nối một hay nhiều tệp OBJ lại với nhau thành một tệp chương trình chạy được với đuôi .EXE.

4. Nếu chương trình gốc viết ra là để dịch ra kiểu .COM thì ta phải dùng chương trình EXE2BIN (đọc là EXEtoBIN) của DOS để dịch tiếp tệp .EXE ra tệp chương trình chạy được với đuôi .COM.

5. Cho chạy chương trình vừa dịch

**3.2. Các cấu trúc lập trình cơ bản**

**3.2.1. Cấu trúc tuần tự**

Cấu trúc tuấn tự là một cấu trúc thông dụng và đơn giản nhất. Trong cấu trúc này các lệnh được sắp xếp tuần tự, lệnh này kế tiếp lệnh kia. Sau khi thực hiện xong lệnh cuối cùng của cấu trúc thì công việc phải làm cũng được hoàn tất.

Ngữ pháp:

Lệnh 1

Lệnh 2

...

Lệnh n

**3.2.2. Cấu trúc IF-THEN**

Cấu trúc:

IF Điều kiện THEN công việc.

Từ ngữ pháp của cấu trúc IF-THEN ta thấy nếu thoả mãn Điều kiện thì Công việc được thực hiện nếu không Công việc sẽ bị bỏ qua. Điều này tương đương với việc dùng lệnh nhảy có điều kiện để bỏ qua một thao tác náo đó trong chương trình hợp ngữ.

**3.2.3. Cấu trúc IF-THEN-ELSE**

Cấu trúc

IF ĐiềuKiện THEN CôngViệc1 ELSE CôngViệc2

Từ ngữ pháp của cấu trúc IF-THEN-ELSE, nếu thoả mãn Điều\_kiện thì Côngviệc1 được thực hiện nếu không thì Côngviệc2 được thực hiện. Điều này tương đương với việc dùng lệnh nhảy có điều kiện và không điều kiện để nhảy đến các nhãn nào đó trong chương trình.

**3.2.4. Cấu trúc SWITCH CASE**

CASE Biểuthức

Giátrị1: Côngviệc1

Giátrị2: Côngviệc2

. . .

GiátrịN: CôngviệcN

END CASE

Từ ngữ pháp của cấu trúc ta thấy nếu Biểuthức có Giátrị1 thì Côngviệc1 được thực hiện. Nếu Biểuthức có Giátrị2 thì Côngviệc2 được thực hiện và cứ tiếp tục cho đến CôngviệcN. Điều này tương đương với việc dùng các lệnh nhảy có điều kiện và nhảy không điều kiện để nhảy các nhãn nào đó trong chương trình hợp ngữ. Cấu trúc CASE có thể thực hiện bằng các cấu trúc lựa chọn lồng nhau.

**3.2.5. Cấu trúc lặp FOR**

FOR Số lần lặp DO Công việc

Từ ngữ pháp cuả cấu trúc FOR - DO ta thấy ở đây Công\_việc được thực hiện lặp đi lặp lại tất cả Số lần lặp lại. Điều này hoàn toàn tương đương với việc dùng lệnh LOOP trong hợp ngữ để lặp lại CX lần một Công việc nào đó, trước đó ta phải gán Số lần lặp cho thanh ghi CX.

**3.2.6. Cấu trúc lặp WHILE**

WHILE Điều kiện DO Công việc

Từ ngữ pháp của cấu trúc WHILE - DO ta thấy: Điều kiện được kiểm tra đầu tiên. Công việc được lặp đi lặp lại chừng nào Điều kiện còn đúng. Điều này trong hợp ngữ hoàn toàn tương đương với việc dùng lệnh CMP để kiểm tra Điều kiện và sau đó dùng lệnh nhảy có điều kiện để thoát khỏi vòng lặp.

**3.2.7. Cấu trúc lặp Repeat-Until**

REPEAT Công việc UMTIL Điều kiện

Từ dạng của cấu trúc REPEAT - UNTIL ta thấy: Công việc được thực hiện đầu tiên. Điều đó có nghĩa là công việc được thực hiện ít nhất một lần. Điều kiện được kiểm tra sau đó. Công việc được lặp đi lặp lại cho tới Điều kiện được thoả mãn. Điều này trong hợp ngữ hoàn toàn tương đương với việc dùng lệnh CMP để kiểm tra Điềukiện và sau đó dùng lệnh nhảy có điều kiện để thoát khỏi vòng lặp.