Cơ bản về lập trình hợp ngữ

TS Nguyễn Hồng Quang

Ví dụ chương trình hợp ngữ 8051

```
;*****************************************************************************
;
; RESET                         ;reset routine
    ORG   0H                    ;locate routine at 00H
    AJMP   START                ;jump to START
;*****************************************************************************
;
; INTERRUPTS  (not used)        ;place interrupt routines at appropriate
                                ;memory locations
    ORG   03H                   ;external interrupt 0
    RETI
    ORG   0BH                   ;timer 0 interrupt
    RETI
    ORG   13H                   ;external interrupt 1
    RETI
    ORG   1BH                   ;timer 1 interrupt
    RETI
    ORG   23H                   ;serial port interrupt
    RETI
    ORG   25H                   ;locate beginning of rest of program
;*****************************************************************************
INITIALIZE:                     ;set up control registers
    MOV   TCON,#00H
    MOV   TMOD,#00H
    MOV   PSW,#00H
    MOV   IE,#00H               ;disable interrupts
    RET
```

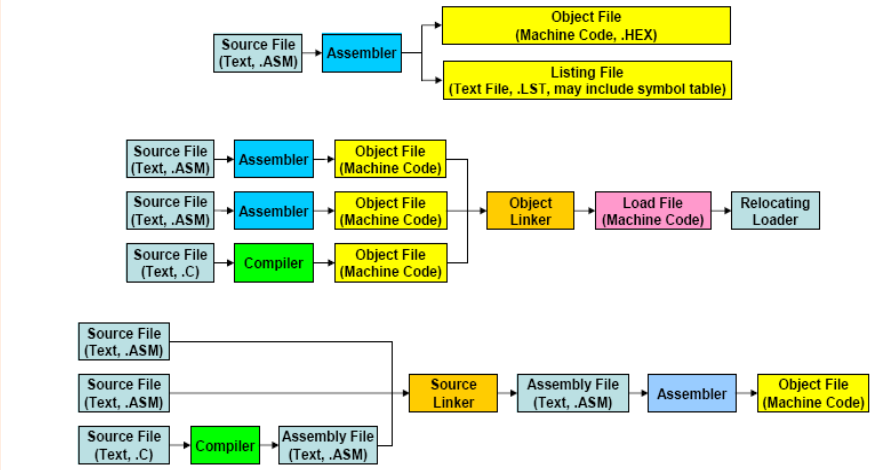
Tổng quan về lệnh trong máy tính

- Các mức lập trình phần mềm máy tính
  – Mã máy là mức thấp nhất thường biểu diễn dưới dạng hex file, lưu dưới dạng .hex, .ihx, .bin
  – Hợp ngữ là ngôn ngữ tương ứng với từng loại Vi xử lý và có thể đọc được dưới dạng tiếng Anh, lưu dưới dạng .asm
  – Ngôn ngữ cấp cao nhằm mục tiêu chuyển chương trình thành dạng có thể đọc được như văn bản tiếng Anh. Ngôn ngữ này cho phép chương trình có thể chạy trên nhiều loại vi điều khiển khác nhau. Ví dụ file .c, .pas, cs, .bas

Hợp ngữ (Assembly)

- Vi xử lý nào cũng có tập lệnh đi theo
- Các lệnh biểu diễn dưới dạng mã nhị phân và được viết dưới dạng gợi nhớ (mnemonics)
- Chu kỳ thực hiện lệnh thường bắt đầu như sau:
  – Vi xử lý bắt đầu reset, lưu bảng vecto ngắt và đọc dữ liệu từ địa chỉ chương trình (kích hoạc chân /PSEN 8051)
  – Tùy thuộc vào mã lệnh, vi xử lý bắt đầu đọc tiếp các toán tử tương ứng và xử lý lệnh nhận được
  – Kết quả có được lại được lưu vào vùng dữ liệu tương ứng
  – Bắt đầu quá trình đọc lệnh tiếp theo cho đến khi kết thúc
- Mỗi lệnh thường có số chu kỳ lệnh và thời gian thực hiện tùy theo từng loại

Quá trình tạo ra mã máy



Ví dụ về file .lst, .hex

```
DUNFIELD 8051 ASSEMBLER: test3                                  PAGE: 1

0000                       1  ********************************************************************************
0000                       2  * EMILY test program: Init data memory with value in A
0000                       3  * Use the 'C'hange register command to set the initial value in A
0000                       4  ********************************************************************************
0800                       5            ORG   $0800
0800  90 00 00             6  BEGIN     MOV   DPTR,#0     Begin at zero
0803  7A 00                7            MOV   R2,#0       Low counter
0805  7B 00                8            MOV   R3,#0       High counter
0807  F0                   9  WRMEM     MOVX  [DPTR],A    Write the value
0808  A3                  10            INC   DPTR        Advance
0809  DA FC               11            DJNZ  R2,WRMEM    Loop 256 times
080B  DB FA               12            DJNZ  R3,WRMEM    LOOP 256*256 times
080D                      13  * Insert the ILLEGAL opcode to halt the simulation
080D  A5                  14            DB    $A5         Halt emily
```

```
:0E0800009000007A007B00F0A3DAFCDBFAA582
:00000001FF
```

Tập thanh ghi

Các thanh ghi chính

– A, B, R0 to R7 : thanh ghi 8 bit

– DPTR : [DPH:DPL] thanh ghi 16 bit

– PC : Bộ đếm chương trình (Instruction Ptr) 16bits

– 4 tập thanh ghi từ (R0-R7)

– Thanh ghi con trỏ ngăn xếp SP

– Thanh ghi trạng thái chương trình PSW (Program Status Word (Flags))

Carry CY, Aux Carry AC, Reg Bank selector, Overflow, Parity

– Thanh ghi đặc biệt (SFRs)

Bộ định thời (Timers), Ngắt (Interrupt) điều khiển vào ra nối tiếp (serial), nguồn

Hợp ngữ – nét cơ bản nhất (assembly)

[nhãn:] [từ gợi nhớ] [các toán hạng]
[; chú giải]

Giá trị luôn có dấu thăng đứng trước #

– #55, #32 etc

Số Hex thì kết thúc bằng chữ H

– #55H, #32H

Thường khi với số bắt đầu bằng chữ #0FFH, #0C1H, #0D2H

- Ví dụ lệnh cơ bản nhất : No operation : NOP !

Ví dụ về cơ bản

Label:
Label: mov A, #25h
Label:
mov A, #5Fh

```
MOV A,#30
MOV A,#11110B
MOV A,#1EH
MOV A,#36Q
```

```
MOV A,#'C'
MOV A,#43H
```

```
MOV A,#'C'       ;Single character
MOV A,#"STRING"  ;String - ERROR!
```

Thanh ghi A

Sủ dụng thường xuyên với lệnh *mov*

Ví dụ về lệnh thường dùng nhất

- mov A, R0
- Mã máy (Opcode) : E8

Ngoài ra trong thanh ghi có trong lệnh khác như

- Instruction : push ACC
- Mã máy : C0 E0

Thanh ghi A, B

- **ACC (Accumulator, Addresses E0h, Bit-Addressable):** Dùng lưu trữ các giá trị trung gian **MOV A,#20h -> MOV E0h,#20h**.
- **B (B Register, Addresses F0h, Bit-Addressable):** Sử dụng trong các phép nhân và chia.

Ví dụ

- MUL AB, nhân 2 số 8 bít trong A, B và lưu kết quả 16, A chứa byte thấp, B chứa byte cao
- DIV AB, chia A bởi B, kết quả lưu vào A, dư lưu vào B

Tập thanh ghi R0-R7

- R0, R1, … R7 dùng làm thanh ghi trung gian
- Có thể có 4 banks
- Chon Bank nào tùy thuộc vào phần mềm, cụ thể là sử dùng bit RS1:RS0 bits trong PSW
- Ngầm định là bank 0

Ví dụ cụ thể

```
ORG   0H          ; Bắt đầu (origin) tại ngăn nhớ 0
MOV   R5, #25H    ; Nạp 25H vào R5
MOV   R7, #34H    ; Nạp 34H vào R7
MOV   A, #0       ; Nạp 0 vào thanh ghi A
ADD   A, R5       ; Cộng nôi dụng R5 vào A (A = A + R5)
ADD   A, R7       ; Cộng nội dung R7 vào A (A = A + R7)
ADD   A, #12H     ; Cộng giá trị 12H vào A (A = A + 12H)

HERE: SJMP HERE   ; ở lại trong vòng lặp này
```

Mã máy ví dụ trên

Địa chỉ ROM	Ngôn ngữ máy	Hợp ngữ
0000	7D25	MOV R5, #25H
0002	7F34	MOV R7, #34H
0004	7400	MOV A, #0
0006	2D	ADD A, R5
0007	2F	ADD A, R7
0008	2412	ADD A, #12H
000A	800A	HERE: SJMP HERE

Thanh ghi DPTR (Data pointer)

- Được dùng để truy xuất bộ nhớ RAM ngoài
- Sử dụng 2 thanh ghi 8 bít để tạo địa chỉ 16 bit
- Chỉ có lệnh tăng DPTR, không có lệnh giảm
- 82 H (DPL), 83H (DPH)

Ví dụ DPTR

MOV A, #55 H

MOV DPTR, # 1000H

MOVX @DPTR, A

- Chương trình chuyển số liệu từ thanh ghi A, ra địa chỉ 1000H

Con trỏ chương trình (PC)

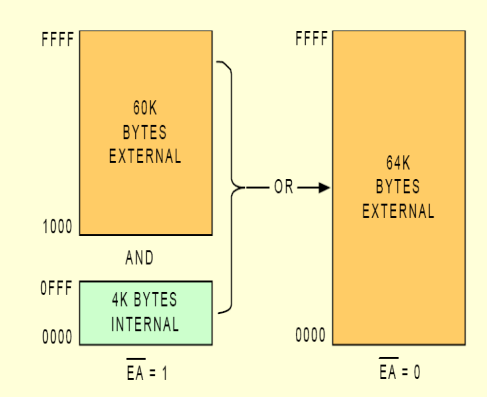
- **The Program Counter (PC)** Con trỏ 2 byte để chỉ chương trình tiếp theo ở lệnh nào
- PC = 0000h khi khởi động
- PC tăng 1,2, 3 byte tùy theo lệnh cụ thể
- Không thể đọc trực tiếp giá trị PC
- Không thể PC=2430h nhưng có thể thực hiện lệnh tương đương LJMP 2430h

Con trỏ ngăn xếp – stack pointer(SP)

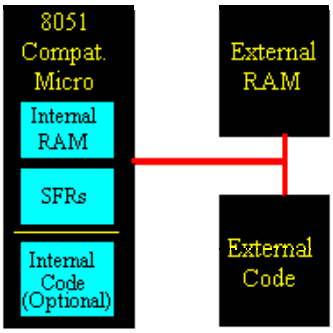
- Dùng để trỏ vị trí tiếp theo khi lấy giá trị ra khỏi ngăn xếp
- Giá trị mặc định là 07h
- Khi sử dụng lệnh PUSH, tự động tăng lên 1
- Các lệnh làm việc với stack
  - PUSH, POP, ACALL, LCALL, RET, and RETI
  - Quan tâm tới SP khi sử dụng ngắt

Kết cấu bộ nhớ

- Bộ nhớ chương trình
  - Chứa trong ROM, hoặc RAM
  - Giới hạn 64 Kbyte
- Bộ nhớ ngoài External RAM
  - Static, flash RAM
  - Giới hạn 64 Kbyte

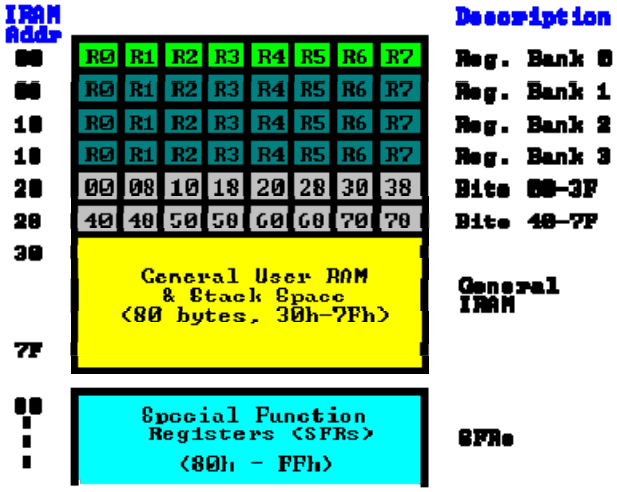


Tổ chức bộ nhớ trong 8051



- Bộ nhớ ngoài (External code memory)
- Bộ nhớ RAM (External RAM)
- Bộ nhớ trên chip (On chip memory)

Kết cấu bộ nhớ On chip



Ram trong

- Dung lượng 128 bytes
- Chia làm 3 phần
  - Register banks
  - Vùng RAM đa mục đích
  - Vùng RAM dùng mục đích chuyên dụng

Memory Type	Start	End	Signal	Instruction
Internal RAM	00H	7FH		MOV A, xxH
External RAM	0000H	FFFFH	RD, WR	MOVX A, @DPTR
External ROM	0000H	FFFFH	PSEN	MOVC, MOVX
Internal ROM	0000H	????H		MOVC

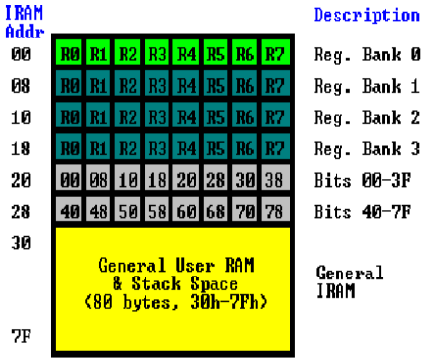
Register Bank

- Phép cộng dữ liệu
  - ADD A, R4
- Tương đương với
  - ADD A, 04H, với mặc định thanh ghi bắt đầu từ địa chỉ 00H
  - Việc thay đổi mặc định quyết định bởi phần mềm
- Register Bank được quyết định bởi bit RS0 và RS1 trong thanh ghi trạng thái PSW

```
MOV PSW,#00h        ;Sets register bank 0
MOV PSW,#08h        ;Sets register bank 1
MOV PSW,#10h        ;Sets register bank 2
MOV PSW,#18h        ;Sets register bank 3
```

Bit memory

- Khoảng giá trị từ 20h – 3Fh, 40h-7Fh, 16 bytes, 128 bit
- Lệnh làm việc bit:
  - **SETB 24h**
  - CLR 24h
- **MOV 20h,#0FFh** -> SETB 0H, SETB 1H, ...,SETB7H



Lưu ý khi làm việc với bit

- SP mặc định là 07h, nếu cần sử dụng băng thanh ghi khác thì cần phải định nghĩa lại SP tại vị trí cao hơn, tương tự với vùng của làm việc bit
- Bit 80h trở lên là vùng thanh ghi đặc biệt (SFR), ví dụ:
  - MOV P0, #02h tương đương với SETB 81h

(Vùng nhớ đặt biệt) SFR

- Bộ nhớ RAM trong khoảng 80H – FFH (128byte)
- Chỉ có 21 thanh ghi hợp lệ
- Thực hiện các chức năng phục vụ riêng cho 8051
- Làm việc như làm việc với bộ nhớ RAM bình thường

Bảng SFR

88	P0	SP	DPL	DPH				PCON
90	TCON	TMOD	TL0	TL1	TH0	TH1		
98	P1							
98	SCON	SBUF						
A8	P2							
A8	IE							
B8	P3							
B8	IP							
C8								
C8								
D8	PSW							
D8								
E8	ACC							
E8								
F8	B							
F8								

87		
8F		
97		
9F		
A7		
AF		
B7		
B9		
C7		
CF		
D7		
DF		
E7		
EF		
F7		
FF		

Blue background are I/O port SFRs
Yellow background are control SFRs
Green background are other SFRs

Lưu ý

- Các thanh ghi theo cột dọc thứ nhất đều có thể làm việc theo bit
- Các thanh ghi SFR còn lại bắt buộc làm việc theo byte
- 3 loại thanh ghi SFR
  - Cổng vào ra
  - Điều khiển
  - Các thanh ghi khác

Các cổng vào ra I/O

- **P0 (Port 0, Address 80h, Bit-Addressable):**
  - Bit 0 của cổng tương ứng với chân P0.0
  - Bit 7 của cổng tương ứng với chân P0.7
  - SETB 80.0 b <-> SETB P0.0
- **P1 (Port 1, Address 90h, Bit-Addressable)**
- **P2 (Port 2, Address A0h, Bit-Addressable)**
- **P3 (Port 1, Address B0h, Bit-Addressable)**

Lưu ý

- Nếu sử dụng RAM ngoài thì các cổng P0, P2 dùng vào tạo dữ liệu địa chỉ
- Cổng P3 có thể dùng cho mục đích đặc biệt khác

Stack pointer (con trỏ ngăn xếp)

- **SP (Stack Pointer, Address 81h):**
- Con trỏ chỉ địa chỉ tiếp theo ngắn xếp
- Ngầm định con trỏ là 07H
  - Nếu có lệnh PUSH, con trỏ tự động tăng lên 1, SP + 1;
- Các lệnh dùng con trỏ SP là: PUSH, POP, LCALL, RET, RETI và ngắt

PCON (power control register)

- **PCON (Power Control, Addresses 87h)**
- **Sử dụng để đặt 8051 ở trạng thái Sleep, tiết kiệm năng lượng**
  - **RAM giữ nguyên giá trị**
  - **Mạch dao động ngừng lại**
  - **ALE, PSEN giữ ở mức không tích cực**
  - **Vcc chỉ cần giá trị 2 V**

PSW (program status word)

- **PSW (Program Status Word, Addresses D0h, Bit-Addressable):** Lưu trữ các cờ làm việc của 8051,

CY	AC	F0	RS1	RS0	OV	–	P

PSW.7 Cờ nhớ CY Carry Flag, khi thực phép số học
PSW.6 Cờ
PSW.5, PSW.1 Dành riêng
PSW.4, PSW.3 -> Chọn dãy thanh ghi tích cực
PSW.3 Cờ tràn OV
PSW.0 P Parity Flag –Cờ chẵn lẻ, xác định số bit lẻ trong thanh chứa A, P =1 nếu A có một số lẻ các bit 1

PSW tiếp

1. Cờ nhớ CY: Khi có nhớ ở bit D7, cờ này thiết lập sau lệnh cộng hoặc trừ 8 bit, có thể lên 1 hoặc xoá về 0 bằng lệnh "SETB C" hoặc "CLR C"
2. Cờ AC: Cờ này báo có nhớ từ bit D3 sang D4 trong phép cộng ADD hoặc trừ SUB. Dùng trong phép tính số học BCD
3. Cờ chẵn lẻ P: Cờ chẵn lẻ chỉ phản ánh số bit một trong thanh ghi A lỡ chẵn hay lẻ. Nếu thanh ghi A chứa một số chẵn các bit một thì P = 0. Do vậy, P = 1 nếu A có một số lẻ các bit một.
4. Cờ tràn OV; Cờ này được thiết lập mỗi khi kết quả của một phép tính số có dấu quá lớn tạo ra bit bậc cao làm tràn bit dấu

Ví dụ: Lệnh ADD và PSW

```
   38          0011 1000
+  2F          0010 1111
---------      ----------
   67           0110 0111
---------       ----------
   -
CY = 0
AC = 1
P = 1
```

```
MOV   A, #38H
ADD  A, #2FH          ; Sau khi cộng A = 67H, CY = 0
```

Ví dụ PSW tiếp

	9C	10011100
+	64	01100100
	100	00000000

Cờ CY = 1 vì có nhớ qua bit D7
Cờ AC = 1 vì có nhớ từ D3 sang D4
Cờ P = 0 vì thanh ghi A không có bit 1 nào (chẵn)

	88	10001000
+	93	10010011
	11B	00011011

Cờ CY = 1 vì có nhớ từ bit D7
Cờ AC = 0 vì không có nhớ từ D3 sang D4
Cờ P = 0 vì số bit 1 trong A là 4 (chẵn)

Thanh ghi thời gian

- **TCON (Timer Control, Addresses 88h, Bit-Addressable):** Xác định các thức làm việc của bộ định thời, bật tắt, ngắt ...
- **TMOD (Timer Mode, Addresses 89h):** Chế độ làm việc 8 bit, 16 bít ..
- **TL0/TH0 (Timer 0 Low/High, Addresses 8Ah/8Ch): Timer 0, giá trị bộ đếm**
- **TL1/TH1 (Timer 1 Low/High, Addresses 8Bh/8Dh):** Timer 1, **giá trị bộ đếm**

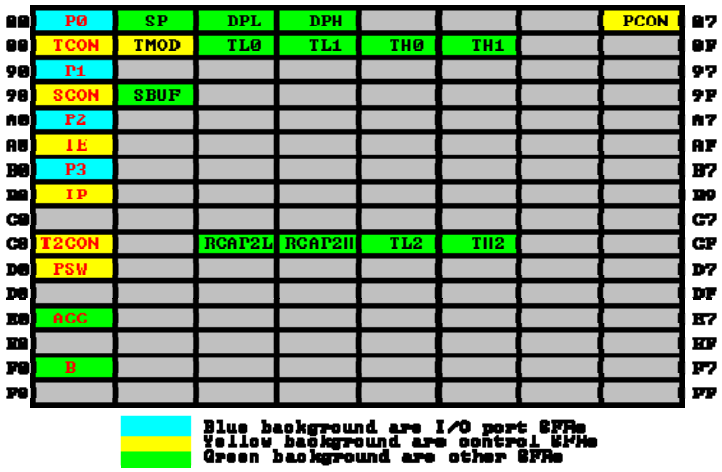
Cổng nối tiếp

- **SCON (Serial Control, Addresses 98h, Bit-Addressable):** Các giá trị khởi đầu cho làm việc với cổng nối tiếp
- **SBUF (Serial Control, Addresses 99h):** Dữ liệu trao đổi giưa vi điều khiển và thiết bị ngoại vi qua cổng nối tiếp

Ngắt

- **IE (Interrupt Enable, Addresses A8h):** Cho phép và không cho phép ngắt
- **IP (Interrupt Priority, Addresses B8h, Bit-Addressable):** Xác định mức độ ưu tiên giữa các ngắt

Bộ nhớ SFR của 89c52



Lệnh nhảy

LJMP new_address

---

new_address:

Mov,...

Nhãn (label), được dịch bởi chương trình dịch

Các lệnh nhảy

- SJMP, +-128 bytes giới hạn
- AJMP, 2 kbytes block giới hạn
- LJMP – 3 bytes

Lênh gọi chương trình

- LCALL
  - Gọi chương trình con theo tên
  - Lưu trữ PC vào stack
- RET
  - Kết thúc chương trình con
  - Lấy giá trị PC từ stack

Thời gian và chu kỳ lệnh

- Chu kỳ lệnh là thời gian tối thiểu để thực hiện một lệnh 1 byte
- Đối với 8051, 12 xung clock thực hiện 1 chu kỳ lệnh
- Nếu tần số 12 MHZ, chu kỳ lệnh 1/1000.000 giây
- Các lệnh toán học yêu cầu 2-3 chu kỳ lệnh

Ví dụ

- Hãy tìm độ trễ thời gian cho chương trình con sau. Giả thiết tần số dao động thạch anh là 11.0592MHz.

			Số chu kỳ máy
DELAY:	MOV	R3, #250	1
HERE :	NOP		1
	NOP		1
	NOP		1
	NOP		1
	DJNZ	R3, HERE	2
	RET		1

Ví dụ định thời

Thời gian trễ bên trong vòng lặp là

[250 (1 + 1 + 1 + 1 + 1 + 2)] x 1.0851µs = 1627.5µs. Cộng thêm hai lệnh ngoài vòng lặp ta có 1627.5µs + 2 x 1.085µs = 1629.67µs.

11.0592/12 = 921.6kHz; Chu kỳ máy là 1/921.6kHz = 1.085µs (micro giây)

Các lệnh phụ trợ trình biên dịch

Lệnh chuyển hướng trong hợp ngữ

– ORG xxxxH : bắt đầu tại xxxxH

– EQU : định nghĩa giá trị

count EQU 25

– DB : define byte, defines data

DATA1: DB 28

DATA2: DB “hello world”

– END : end of assembly fil

Các lệnh phụ

- Lệnh ORG: Chỉ lệnh ORG được dùng để báo bắt đầu của địa chỉ. Số đi sau ORG có kể ở dạng Hex hoặc thập phân. Một số hợp ngữ sử dụng dấu chấm đứng trước".ORG" thay cho "ORG".
- lệnh EQU: Lệnh EQU dùng gắn một giá trị hằng số với nhãn dữ liệu sao cho khi nhãn xuất hiện trong chương trình giá trị hằng số của nó sẽ được thay thế đối với nhãn.

```
COUNT EQU 25
MOV R3, #count
```


Các lệnh phụ

Lệnh DB (định nghĩa byte). Lệnh DB dùng để định nghĩa dữ liệu 8 bit. Bất kể ta sử dụng số ở dạng thức nào thì hợp ngữ đều chuyển đổi chúng về thành dạng Hex.

Lệnh DB là lệnh mà có thể được sử dụng để định nghĩa các chuỗi ASCII lớn hơn 2 ký tự.

```
        ORG   500H
DATA1:  DB    2B              ; Số thập phân (1C ở dạng Hex)
DATA2:  DB    00110101B       ; Số nhị phân (35 ở dạng Hex)
DATA3:  DB    39H             ; Số dạng Hex
        ORG   510H
DATA4:  DB    "2591"          ; Các số ASCII
        ORG   518H
DATA5   DB    "My name is Joe"  ; Các ký tự ASCII
```

Lệnh phụ

- DW: define word (định nghĩa từ dữ liệu), cách dùng tương tự DB
  - temp: DW 'A', 1342H,
- DS (define storage)
  - Length EQU 25H
  - Buffer: DS Length
- END, .END dùng báo cho hợp ngữ kết thúc quá trình dịch tại thời điểm nhận lệnh

Ví dụ

```
Begin:
                Mov A,# 0           ; (A) = 0
                MOV P1,A            ; Gửi ra cổng P1
                Call    wait        ; Gọi chương trình tạo trễ
                Mov A,# 255     ; (A) =255
                Mov P1,A            ; Gửi  ra cổng P1
                Call    Wait        ; Gọi chương trình tạo trễ
                Jmp  Begin          ; Nhảy trở lại Begin
Wait:
                Mov R7, #255    ; (R7)=255 - Số đếm vòng ngoài
   Schl 1:      Mov R6, #255    ;(R6)= 255- Số đếm vòng trong
   Schl 2:      Djnz R6, Schl 2 ;Nếu (R6) ≠ 0 thi quay lại Schl 2
                Djnz R7, Schl 1 ; Nếu (R7) ≠ 0 thi quay lại Schl 1
                Ret
.End
```