IT4778
LẬP TRÌNH HỆ THỐNG

TS. Đỗ Quốc Huy
huydq@soict.hust.edu.vn

Ngôn ngữ lập trình C

Nội dung chính

1. Ngôn ngữ lập trình C
2. Kiểu dữ liệu có cấu trúc
3. Con trỏ và cung cấp nhớ
4. Vào ra
5. Lập trình hệ thống với Turbo C

Kiểu dữ liệu cơ bản

- Khác nhau giữa các phiên bản ngôn ngữ C
  - Borland C++/**Turbo C++**/ Microsoft C /gcc,..
- Các kiểu dữ liệu cơ bản
  - (unsigned) char, int (short, long)
  - float, double
- C chuẩn không có các kiểu BYTE/ WORD…
  - typedef unsigned char **BYTE**;
  - typedef unsigned int **WORD**; //TC: int 2bytes
  - typedef long **DWORD**

Kiểu dữ liệu cơ bản

- **unsigned/signed**: Đánh dấu số không dấu/có dấu
- **C99** thêm một số kiểu nguyên: #include<stdint.h>
  - int8_t
  - int16_t
  - int32_t
  - int64_t
  - uint8_t
  - uint16_t
  - uint32_t
  - uint64_t

Type	X86 (32 bit)	X64 (64bit)	printf
char	1	1	%c
short [int]	2	2	%d, %ud
int	4	4	%d, %ud
long	4	4/8	%ld
long long [int]	8	8	%lld %I64d
float	4	4	%f
double	8	8	%lf
long double	12	16	%Lf
Pointer	4	8	%p
void	1		

Kiểu dữ liệu cơ bản

```
#include <stdio.h>
#include <stdint.h>
int main() { // Stored in text
    printf("size of char : %d\n",sizeof(char));
    printf("size of short : %d\n",sizeof(short));
    printf("size of int : %d\n",sizeof(int));
    printf("size of long : %d\n",sizeof(long));
    printf("size of long long : %d\n",sizeof(long long));
    printf("size of float : %d\n",sizeof(float));
    printf("size of double : %d\n",sizeof(double));
    printf("size of long double : %d\n",sizeof(long double));
    printf("size of Pointer : %d\n",sizeof(&main));
    printf("size of void : %d\n\n",sizeof(void));
    printf("size of int8_t : %d\n",sizeof(uint8_t));
    printf("size of int16_t : %d\n",sizeof(uint16_t));
    printf("size of int32_t : %d\n",sizeof(uint32_t));
    printf("size of int64_t : %d\n",sizeof(uint64_t));
    return 0;
}
```

```
size of char :          1
size of short :         2
size of int :           4
size of long :          4
size of long long :     8
size of float :         4
size of double :        8
size of long double:    12
size of Pointer :       4
size of void :          1

size of int8_t :        1
size of int16_t :       2
size of int32_t :       4
size of int64_t :       8
```

```
size of char :          1
size of short :         2
size of int :           4
size of long :          4
size of long long :     8
size of float :         4
size of double :        8
size of long double:    16
size of Pointer :       8
size of void :          1

size of int8_t :        1
size of int16_t :       2
size of int32_t :       4
size of int64_t :       8
```

Biểu thức

- **Biểu thức:** Kết hợp các toán hạng bởi các toán tử
  - Biểu thức số học, biểu thức quan hệ, biểu thức logic
- **Toán tử**

Loại	Toán tử
Số học	+ - * / %
Quan hệ	= != > >= < <=
Logic mệnh đề	&& || !
Bit	& | ^ ~ << >>
Tăng/giảm	a++, ++a a--, --a
Điều kiện	? :
Phẩy,	,
Địa chỉ, nội dung	& *

Toán tử bit → Ví dụ

char Op1 = 83, Op2 = -38, Op = 3;

char r = Op1 & Op2;

0 1 0 1 0 0 1 1

1 1 0 1 1 0 1 0

r = 0 1 0 1 0 0 1 0 → (82)

char r = Op1 | Op2;

0 1 0 1 0 0 1 1

1 1 0 1 1 0 1 0

r = 1 1 0 1 1 0 1 1 → (-37)

char r = Op1 ^ Op2;

0 1 0 1 0 0 1 1

1 1 0 1 1 0 1 0

r = 1 0 0 0 1 0 0 1 → (-119)

char r = ~ Op2;

1 1 0 1 1 0 1 0

r = 0 0 1 0 0 1 0 1 → (37)

unsigned char r = Op1 | Op2; r = 1 1 0 1 1 0 1 1 → **219**

Toán tử bit → Ví dụ

char Op1 = 83, Op2 = -38, Op = 3;

char r = Op1 **>>** Op;

0 1 0 1 0 0 1 1

r = **0 0 0** 0 1 0 1 0 → (10)

char r = Op2 **>>** Op;

1 1 0 1 1 0 1 0

r =**1 1 1** 1 1 0 1 1 → (-5)

unsigned char Op =218;

unsigned char r =Op >> 3;

1 1 0 1 1 0 1 0

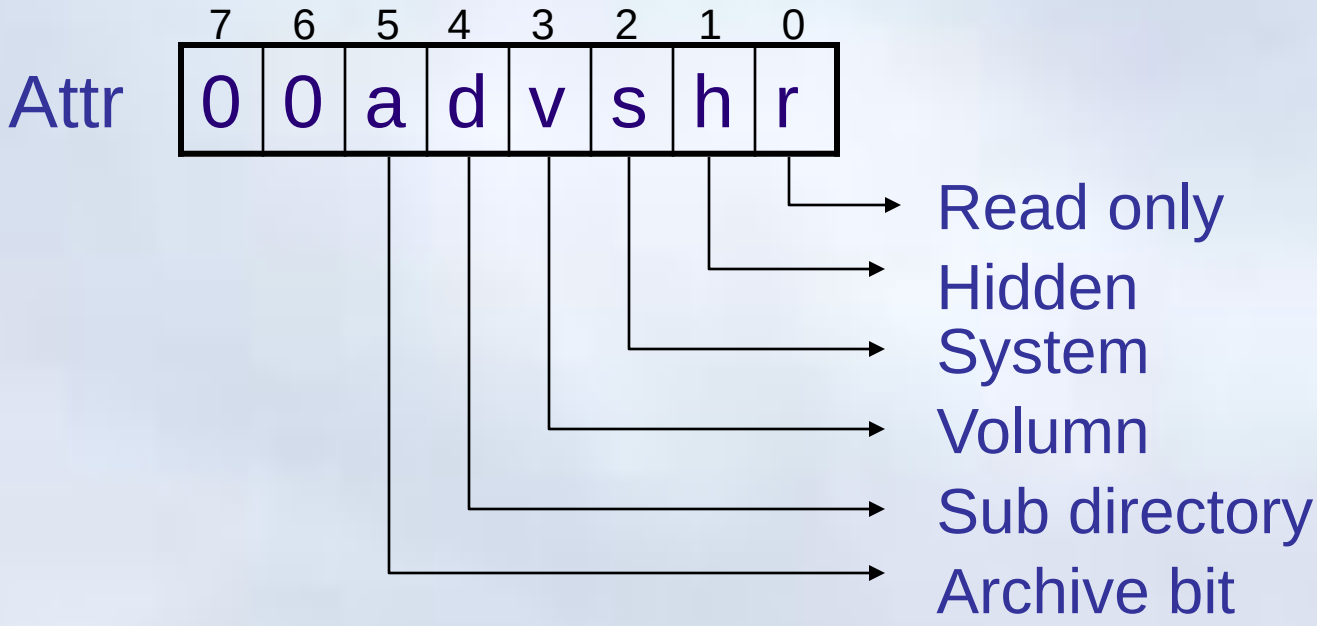
r =**0 0 0** 1 1 0 1 1 → (27)

char r = Op2 **<<** 2;

r = 0 1 1 0 1 0 0 0 →(104)

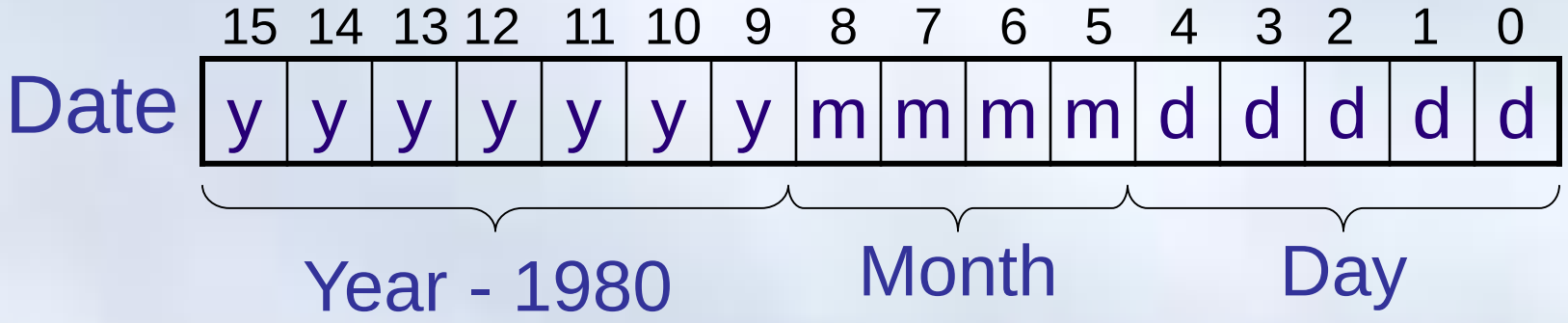
(unsigned) int r = Op2<<2 →?

Toán tử bit → Ứng dụng



- Thuộc tính có thể biểu diễn dưới dạng bit
  - Đọc ghi thuộc tính → Cần làm việc với bit
- Reset bit số 4: Attr &=~16//**16**: 00010000
- Set bit 2, 3: Attr |= 12 // **12**: 00001100

Toán tử bit → Ứng dụng



- Day = Date & 31
- Month = (Date & 0x01E0) >> 5
- Year = (Date & 0x7E00) >> 9 +1980

- Date = Day +(Month<<5)+(Year-1980)<<9

Các cấu trúc lập trình

1. Cấu trúc rẽ nhánh
   - if (bieu_thuc), if (bieu_thuc) … else
   - switch (bieu_thuc) {(case/break/default)}
2. Cấu trúc lặp
   - for(b.thuc_1; b.thuc_2; b.thuc_3) CauLenh;
   - while (bieu_thuc) CauLenh;
   - do Cau_Lenh while (bieu_thuc);
3. Các lệnh thay đổi cấu trúc lập trình
   - continue/ break

Nội dung chính

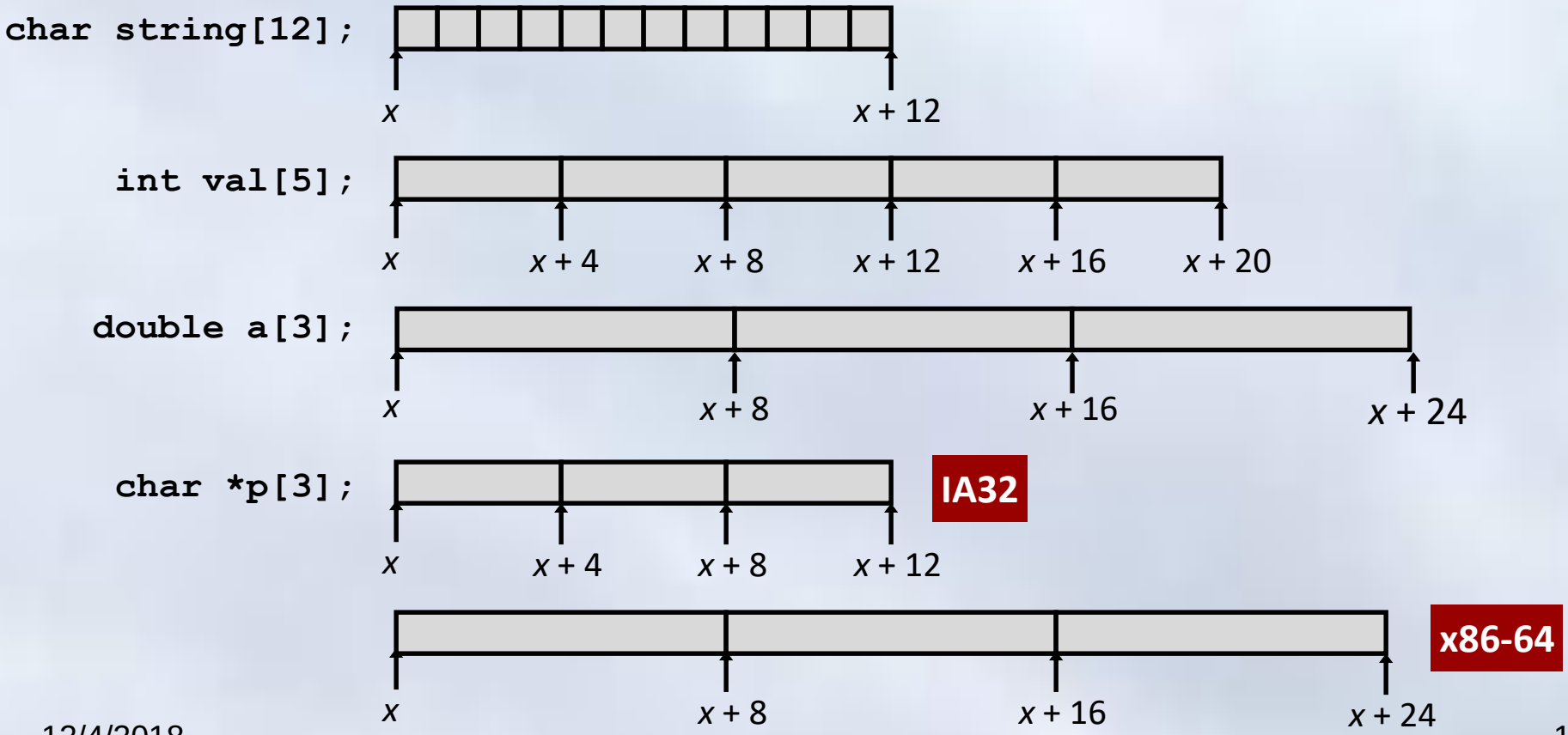
1. Ngôn ngữ lập trình C
2. Kiểu dữ liệu có cấu trúc
3. Con trỏ và cung cấp nhớ
4. Vào ra
5. Lập trình hệ thống với Turbo C

Kiểu dữ liệu có cấu trúc

- **Mảng**
  - Chiếm vùng nhớ liên tục
  - Không kiểm tra phạm vi mảng
- **Xâu:**
  - Chuỗi mã ASCII, kết thúc ký tự null→ASCIIZ string (**Z**: zero)
  - Phù hợp với các hàm của BIOS
  - Thư viện chuẩn **string.h** chứa các hàm xử lý xâu: strcmp,..
- **Cấu trúc**
  - Sắp xếp theo trật tự khai báo
  - Các trường *có thể không liên tục* mà sắp xếp để bắt đầu theo địa chỉ chẵn (WORD Alignement)
- **Hợp nhất (union):**
  - Cho phép xếp các kiểu DL khác nhau trên cùng vùng nhớ

Kiểu mảng

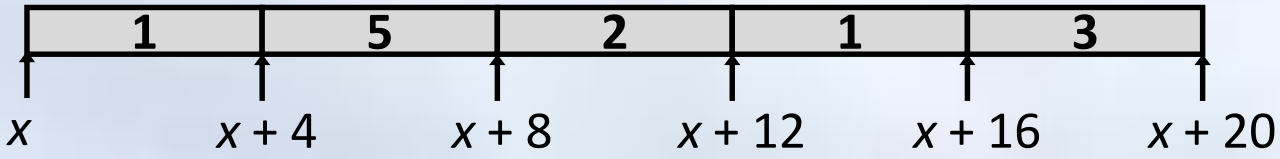
- ***T* A[*L*];**
  - Mảng A độ dài L, các phần tử có kiểu là T
  - Chiếm vùng nhớ liên tục, kích thước ***L*** * **sizeof**(*T*) bytes



Kiểu mảng→Truy nhập phần tử

- ***T* A[*L*];**
  - A là hằng con trỏ, trỏ tới phần tử 0 của mảng

```
int val[5];
```



- Reference Type Value

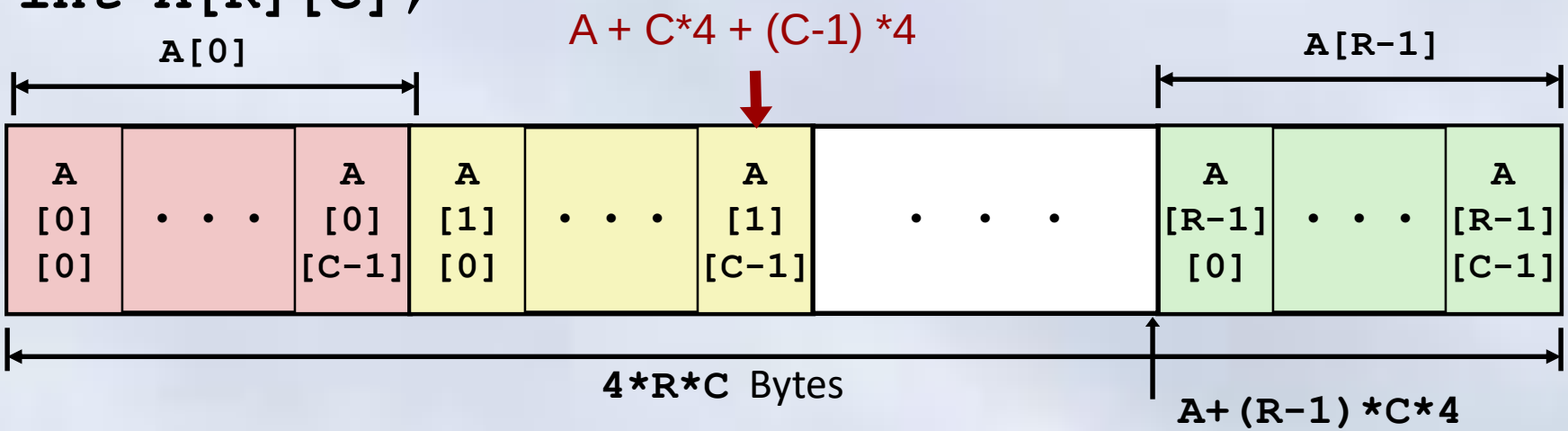
val[4]	**int**	3
val	**int ***	*x*
val+1	**int ***	*x* + 4
&val[2]	**int ***	*x* + 8
val[5]	**int**	??
***(val+1)**	**int**	5
val + *i*	**int ***	*x* + 4 *i*

Kiểu mảng→Mảng nhiều chiều 1 (nested)

- ***T* A[*R*][C];**
  - A là mảng 2 chiều, R dòng, C cột
  - Kích thước R * C * sizeof(T)
  - Lưu trữ theo dòng
  - A[i] là mảng C p/tử, kiểu T
    - Địa chỉ **A + i * C * sizeof(T)**
    - A[i][j] có địa chỉ: **A+ i * C * sizeof(T) +j * sizeof(T)**

```
A[0][0]   • • •  A[0][C-1]
   •                 •
   •                 •
   •                 •
A[R-1][0] • • •  A[R-1][C-1]
```

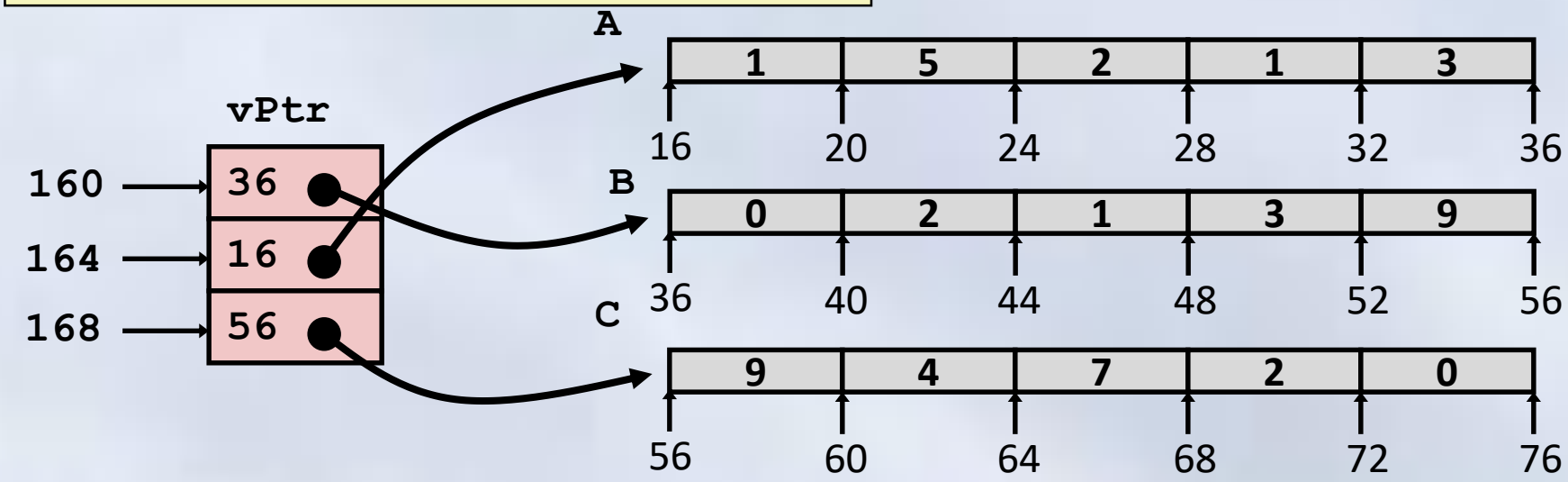
`int A[R][C];`



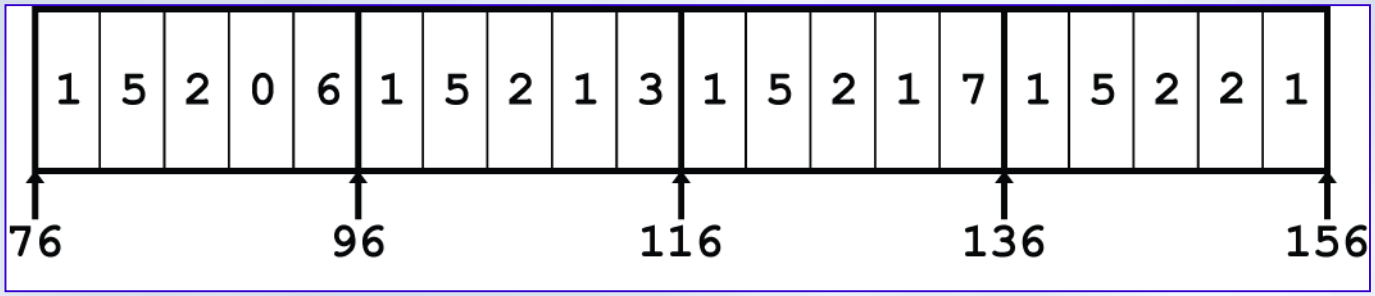
Kiểu mảng→Mảng nhiều chiều 2 (Multi-Level)

```
#define VEC_LEN 5
#define UCOUNT 3

typedef int Vec[VEC_LEN];

Vec A = { 1, 5, 2, 1, 3 };
Vec B = { 0, 2, 1, 3, 9 };
Vec C = { 9, 4, 7, 2, 0 };
int * vPtr[UCOUNT] = {B, A, C};
```

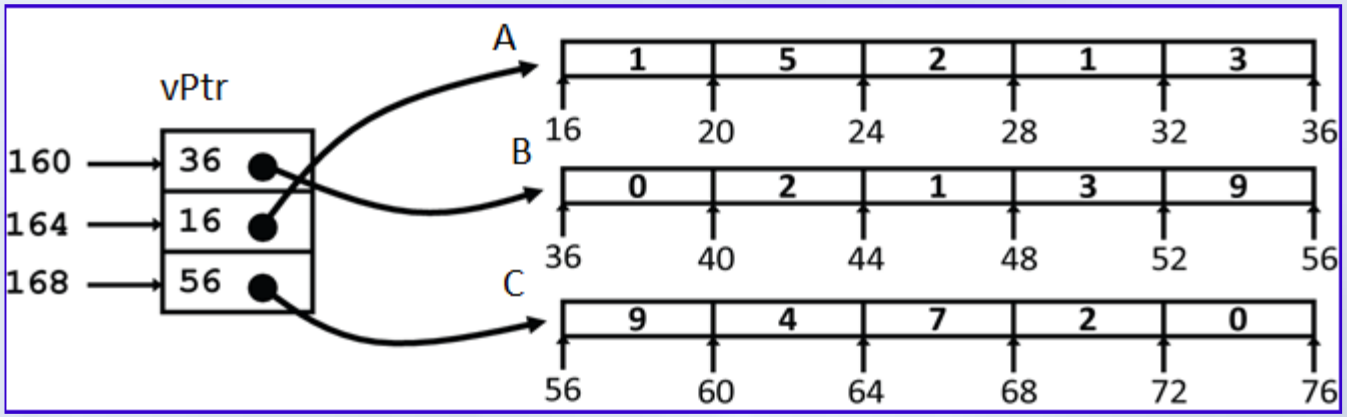
- vPtr là mảng 3 phần tử
- Mỗi p.tử là một con trỏ
  - 4/8 byte
- Mỗi con trỏ trỏ tới một mảng các số int
- Truy nhập vPtr[1][1]=5



Kiểu mảng→Mảng nhiều chiều



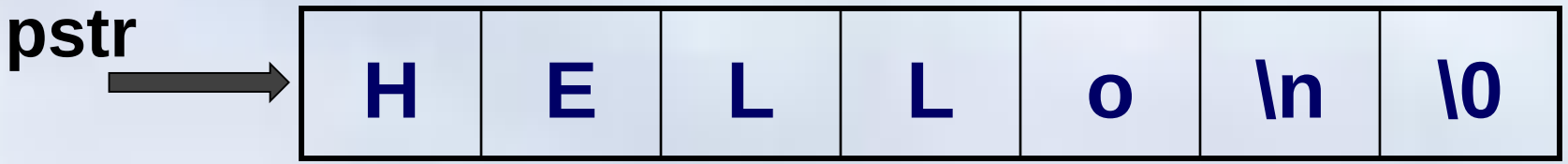
$$A[r][c] = MEM[\ A + 20 * r + 4 * c]$$



$$A[r][c] = MEM[MEM[vPtr+4*r] +4 * c]$$

Xâu

- Xâu là một mảng các ký tự
  - char str[7] = “Hello\n”;
  - char str[] = “Hello\n”;
  - char * pstr =“Hello\n”



- Kích thước của xâu
  - sizeof(): Kích thước khai báo (sizeof(str) =7)
  - strlen(): Độ dài xâu (strlen(str) =6)
- Các hàm xử lý xâu: thư viện string.h
  - strlen(), strcpy(), strcat(), strcmp(), strstr(), strchr(), strrchr(), sprintf(), sscanf(), strtok(),…

Xâu→Buffer overflow

```
#include <stdio.h>
#include <string.h>

int main() {
   char *str1 = "012345";
    char str2[6];
    int i = 0xff;

    printf( "i = %d\n", i );
    strcpy( str2, str1 );
    printf( "i = %d\n", i );
    return 0;
}
```

```
i = 255
i = 0
```

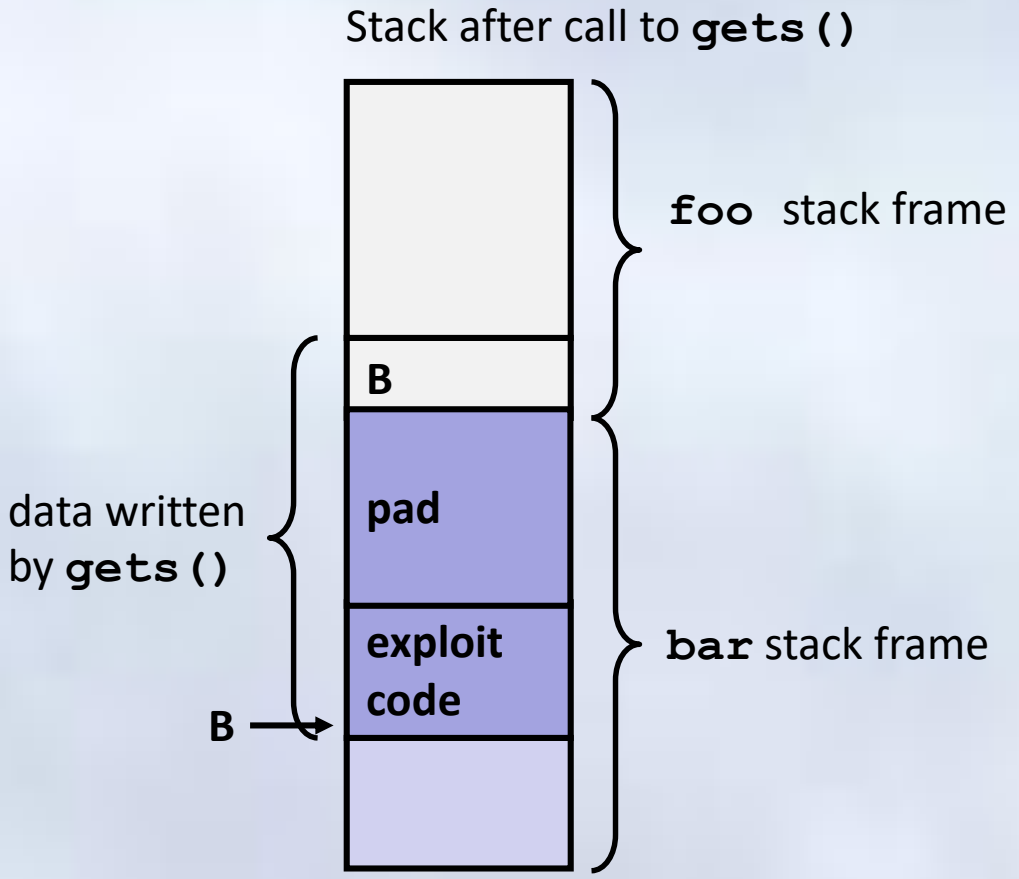
- Các hàm xử lý xâu không kiểm tra kích thước dữ liệu
- Buffer overflow diễn ra khi dữ liệu trong strack bị ghi đè lên
- Nếu ghi đè lên địa chỉ quay trở về, có thể gây nguy hiểm cho hệ thống

Xâu→Buffer overflow

```
void foo(){
  bar();
  ...
}
```

return address A

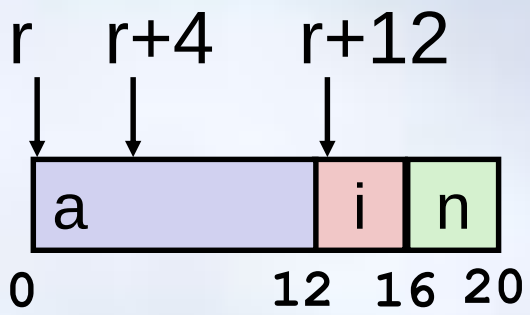
```
int bar() {
  char buf[64];
  gets(buf);
  ...
  return ...;
}
```



- Xâu nhập vào chứa các byte thể hiện mã thực thi
- Ghi đè lên địa chỉ quay về A bởi địa chỉ của buffer B
- Khi `bar()` thực hiện lệnh `ret`, nó sẽ nhay tới mã của B

Cấu trúc

```
struct rec {
  int a[3];
  int i;
  struct rec *n;
} Node;
```



Memory Layout

- Tập hợp các biến rời rạc, có thể khác kiểu
  - Các biến là trường, được truy nhập qua tên
- Chiếm vùng nhớ liên tục
- Con trỏ cấu trúc, trỏ tới byte đầu tiên,
  - Các trường truy nhập qua vị trí tương đối (offset)
  - **Ví dụ:** char * r = (char *)&Node;

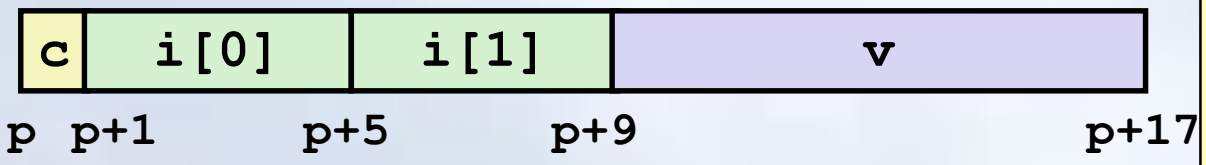
    (int) (r+12) = 100 ⇔ Node.i = 100

Cấu trúc → Alignement

- Kiểu dữ liệu cơ bản (từ máy) kích thước K byte ⇒ địa chỉ phải là bội số của K
- Đòi hỏi trên một số hệ thống (*IA32 là khuyến nghị*)
  - Xử lý khác nhau, theo hệ thống, hệ điều hành
- Sử dụng dữ liệu sắp hàng cho phép truy nhập bộ nhớ hiệu quả hơn
  - Dữ liệu nằm trên 2 từ máy, đọc/ ghi 2 lần
  - Dữ liệu nằm trên 2 từ máy ⇒ có thể nằm trên 2 trang (page) ⇒ không hiệu quả cho bộ nhớ ảo
- Chương trình dịch tự động chèn các byte giữa các trường để đảm bảo nguyên tắc sắp hàng

Cấu trúc → Alignement

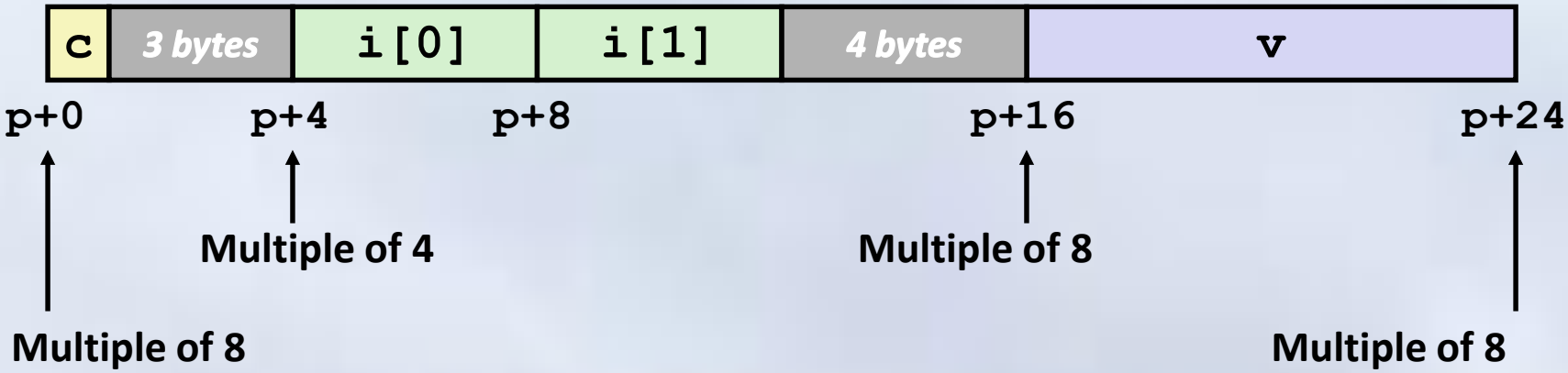
- Unaligned Data

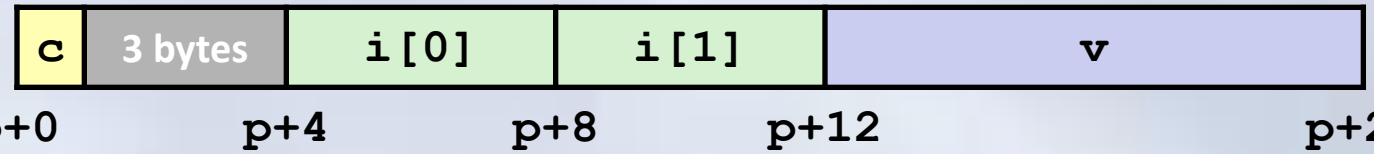



```
struct S1 {
  char c;
  int i[2];
  double v;
} *p;
```

- Aligned Data

x86-64 or IA32 Windows



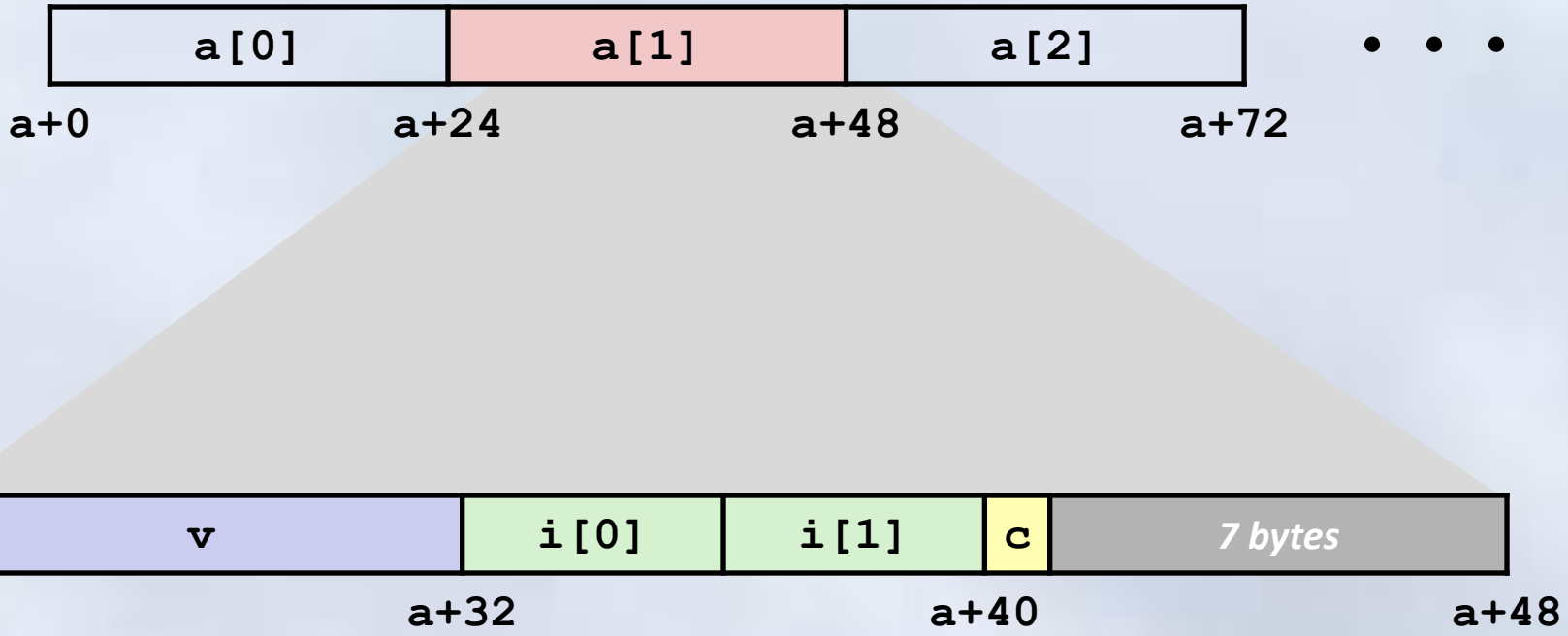


IA32 Linux

Cấu trúc → Mảng cấu trúc

- Mỗi phần tử có kích thước là bội số kiểu DL cơ bản
- Mọi phần tử của mảng đều phải được sắp hàng

```
struct S2 {
   double v;
   int i[2];
   char c;
} a[10];
sizeof(A)=240
```

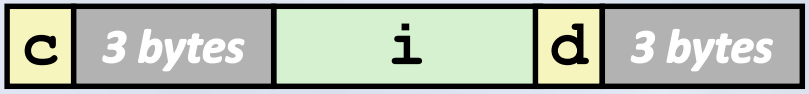


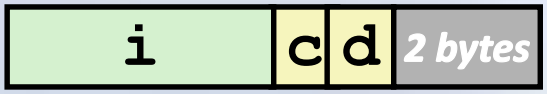
Cấu trúc → Alignement

- Tiết kiệm bộ nhớ:
  - Đặt trường có kích thước lớn lên trước

```
struct S4 {
  char c;
  int i;
  char d;
} *p;
```

```
struct S5 {
  int i;
  char c;
  char d;
} *p;
```

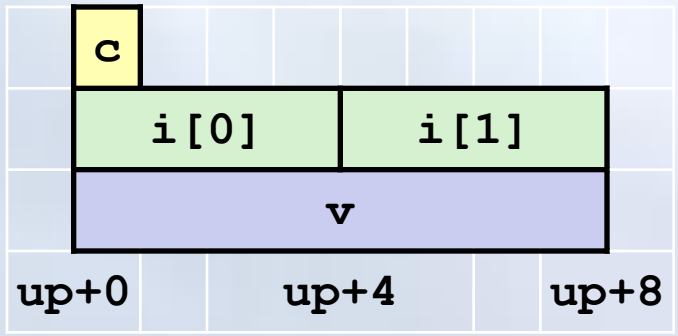




Union

- Cung cấp vùng nhớ bằng yêu cầu của phần tử kích thước lớn nhất
- Chỉ dùng 1 trường tại một thời điểm

```
union U1 {
  char c;
  int i[2];
  double v;
} *up;
```




```
struct S1 {
  char c;
  int i[2];
  double v;
} *sp;
```

Nội dung chính

1. Ngôn ngữ lập trình C
2. Kiểu dữ liệu có cấu trúc
3. Con trỏ và cung cấp nhớ
4. Vào ra
5. Lập trình hệ thống với Turbo C

Kiểu con trỏ

- Dùng để xác định địa chỉ một ô nhớ
- Phụ thuộc chế độ quản lý bộ nhớ
- Chế độ bảo vệ (Windows)
  - Phụ thuộc vào kiến trúc máy, hệ điều hành và chương trình dịch
  - Kích thước 32/64 byte
- Chế độ thực (TC 3.0++) NEAR /FAR
  - int near * Ptr; //Kiểu bộ nhớ SMALL, 16 byte
  - int far * Ptr; // 32 byte

Con trỏ hàm

- Con trỏ trỏ đến một vị trí trong bộ nhớ
- Hàm là đoạn mã lệnh nằm trong bộ nhớ
- Con trỏ trỏ tới lệnh đầu tiên của hàm con trỏ hàm

Ví dụ

- **typedef void (*FuncPtr)(int a);**

FuncPtr là kiểu con trỏ, trỏ tới một hàm có tham số là một số nguyên và trả về kiểu void

Con trỏ hàm→Ví dụ

```
#include<conio.h>
typedef void (*FuncPtr)(int a);
void AddArray( int *array, int n, FuncPtr func ) {
        int i;
        for( i = 0; i < n; i++ )   (*func)( array[ i ] );
}
int s = 0;
void Sum( int val ){  s += val; }
main( ){
        int a[ ] = {3,4,7,8};
        AddArray(a, sizeof(a)/sizeof(int), Sum);
        printf("S=%d", s);
}
```

Nội dung chính

1. Ngôn ngữ lập trình C
2. Kiểu dữ liệu có cấu trúc
3. Con trỏ và cung cấp nhớ
4. Vào ra
5. Lập trình hệ thống với Turbo C

Nội dung chính

- WORD Alignement
- Truy nhập thanh ghi của VXL
- Gọi dịch vụ của hệ thống từ C
- Sử dụng vùng đệm
- Làm việc với cổng vào ra
- Thư viện <dos.h>

WORD Alignement

Addr	Content	Type
00h	Reserved	21B
15h	File attribute	1B
16h	Last modi. Time	1W
18h	Last Modi. Date	1W
1Ah	File Size	1DW
1Eh	File Name & Ext	13B
Total	43 bytes	

Directory Entry Structure
DOS Function 4Eh/4Fh

```
type def struct{
   unsigned char Res[21];
   unsigned char Attr;
   unsigned int Time;
    unsigned int Date;
   unsigned long Size;
   char        Name[13];
}DIR_ENTRY;
```

sizeof(DIR_ENTRY) = 43 ?

Truy nhập thanh ghi của VXL

- Các thanh ghi của VXL được truy nhập qua cấu trúc union REGS, struct SREGS, struct REGPACK
- Chỉ có cấu trúc thanh ghi cơ bản, 16bit
- Cấu trúc REGS chỉ sử dụng cờ Carry

```
struct SREGS{
    unsigned int es;
    unsigned int cs;
    unsigned int ss;
    unsigned int ds;
}
```

```
struct REGPACK{
    unsigned r_ax, r_bx, r_cx,
    r_dx;
    unsigned r_bp, r_si, r_di;
    unsigned r_ds, r_es, r_flags;
}
```

Truy nhập thanh ghi của VXL

```
union REGS{
    struct WORDREGS x;
    struct BYTEREGS h;
}
```

```
struct BYTEREGS{
    unsigned char al, ah;
    unsigned char bl, bh;
    unsigned char cl, ch;
    unsigned char dl, dh;
}
```

```
struct WORDREGS{
    unsigned int ax;
    unsigned int bx;
    unsigned int cx;
    unsigned int dx;
    unsigned int si;
    unsigned int di;
    unsigned int cflag;
}
```

Gọi dịch vụ của hệ thống từ C

- Dịch vụ của BIOS được qua ngắt mềm
  - Dịch vụ của Dos cũng gọi qua ngắt mềm
- Gọi ngắt trong Turbo C
  - Thư viện <dos.h>
  - **intdos(..) / intdosx(..):** Gọi các hàm DOS API
    - DOS API được cung cấp với ngắt mềm 21h
  - **int86(..)/ int86x(..), intr():** Gọi ngắt của VXL intel
  - intdosx() và int86x() có sử dụng thanh ghi đoạn
  - intdos() và int86() chỉ dùng thanh ghi thông dụng

Gọi dịch vụ của hệ thống từ C

- intdos (union REGS * inregs,
  union REGS * outregs)
- intdosx (union REGS * inregs,
  union REGS * outregs, struct SREGS *sregs)
- int86(int intNbr, union REGS * inregs,
  union REGS * outregs)
- int86x(int intNbr, union REGS * inregs,
  union REGS * outregs, struct SREGS *sregs)
- intr(int intNbr, struct REGPACK *regs)

Ví dụ

```
#include<dos.h>
#include<stdio.h>
int main(){
    union REGS R;
    struct SREGS sR;
    R.h.ah = 0x01;      //kich thuoc con tro
    R.x.cx = 0x011F;
    int86x(0x10,&R,&R,&sR);
    R.h.ah = 0x02;              //Dịch vụ đặt lại vi tri con tro
    R.x.dx = 0x0505;            //Dòng 5, cột 5
    int86x(0x10, &R,&R,&sR);
    return 0;
}
```

Sử dụng vùng đệm

- Lưu trữ dữ liệu trong các thao tác vào /ra
- Nhiều dịch vụ BIOS cần con trỏ vùng đệm
  - Con trỏ vùng đệm thuộc loại FAR (Seg,Ofs)
- Truyền con trỏ tới thanh ghi cho lời gọi ngắt
  - Macro **FP_SEG()** và **FP_OFF()**
  - Ví dụ: sR.es = FP_SEG(Buf)

    R.x.dx = FP_OFF(Buf)
- Nhận con trỏ vùng đệm từ lời gọi ngắt
  - Macro **MK_FP()**
  - VD: unsigned char far * p=MK_FP(R.ds,R.x.bx)

Ví dụ 1

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
unsigned A[10];
void main(){
    unsigned * Ptr = (unsigned *) malloc(100);
     printf("Dia chi cua main %X:%X\n", FP_SEG(main), FP_OFF(main));
     printf("Dia chi cua A %X:%X\n", FP_SEG(A), FP_OFF(A));
     printf("Dia chi cua Ptr %04X:%04X\n", FP_SEG(&Ptr), FP_OFF(&Ptr));
     printf("Dia chi Ptr tro den %04X:%04X\n", FP_SEG(Ptr), FP_OFF(Ptr));
     return;
}
```

Ví dụ 2→Xem ngày xuất xưởng máy

- Ghi dưới dạng mã ASCII trong ROM tại địa chỉ FFFF5

```
#include<dos.h>
#include<stdio.h>
int main(){
    char far * Ptr = MK_FP(0xF000,0xFFF5);
    int i;
    for(i = 0; i < 8; i++) printf("%c",Ptr[i]);
   return 0;
}
```

Ví dụ 3→Hiện thị xâu ký tự (hàm 9h, ngắt 21h)

- Chuỗi ASCII kết thúc bởi ký tự '$'
- Cặp thanh ghi DS:DX trỏ tới chuỗi ký tự

```
#include <dos.h>
void main(){
    union REGS R;
    struct SREGS sR;
    char Msg[] = "Hello world!$";
    R.h.ah = 0x09;
    R.x.dx = FP_OFF(Msg);
    sR.ds = FP_SEG(Msg);
    intdosx(&R,&R,&sR);
}
```

Ví dụ 4→Lấy thư mục hiện thời (hàm 47h)

- Cặp thanh ghi DS:DI trỏ tới vùng đệm chứa kết quả
- DL cho biết ổ đĩa (0 ổ hiện tại, 1 ổ A, 2 ổ B,..)

```
#include <dos.h>
#include <stdio.h>
char Str[100];
void main(){
    union REGS R;           struct SREGS sR;
    R.h.ah = 0x47;          R.h.dl = 0;
    R.x.si = FP_OFF(Str);   sR.ds = FP_SEG(Str);
    intdosx(&R,&R,&sR);
    printf("%s",MK_FP(sR.ds,R.x.si));
}
```

Chèn lệnh hợp ngữ

- Thực hiện nhanh do không dùng CALL, RET
- Đơn giản do không sử dụng khái niệm đối tượng bên ngoài (**extern**)
- Cú pháp

```
asm statement
asm{
    statement //chú thích
    ……
    statement
}
```

Chèn lệnh hợp ngữ→Ví dụ

```
void clrscr(){
  asm{
    MOV AX,0600h
    MOV CX,0
    MOV BH,7
    MOV DX,184Fh
    INT 10h     //Xoa màn hình
    MOV AH, 2
    MOV DX, 0
    MOV BH, 0
    INT 10h   //Dat vi tri con tro
  }
}
```

```
void printf(char * str){
    asm{
        MOV AH, 9
        MOV DX, str
        INT 21h
    }
}
int main(){
    char str[] = "Hello world$";
    clrscr();
    printf(str);
    return 0;
}
```

Vào/ra trực tiếp

- #include <dos.h> //Borland
  - int inport(unsigned portID)
  - unsigned char inportb(int portID)
  - void outport(int portID, int value)
  - void outportb(int portID, unsigned char value)
- #include <conio.h> //Microsoft
  - int inp(unsigned portID)
  - unsigned inpw(int portID)
  - void outp(int portID, int dataByte)
  - void outpw(int portID, unsigned dataWord)

Thư viện <dos.h>

- Cung cấp các dịch vụ cho phép làm việc với hệ điều hành MS-DOS
  - Hàm, hằng số, kiểu dữ liệu, biến toàn cục
    - Các hàm gọi các dịch vụ tương ứng của hệ điều hành DOS
- Ví dụ: Thay đổi vector ngắt
  - Bảng vector ngắt gồm 256 vector, trỏ đến 256 chương trình xử lý ngắt ương ứng
  - Thay đổi được địa chỉ một vector ngắt trỏ đến
    - void interrupt (*getvect(int intNbr))();
    - void setvect(int intNbr, void interrupt (*isr) ( ));

    interrupt là từ khóa của C, định nghĩa hàm là thủ tục xử lý ngắt

Thay đổi vector ngắt→Ví dụ

```
#include <dos.h>
#define INTR 0x09
void interrupt ( *oldhandler)();
int count=0;
void interrupt handler(){
    unsigned al = inportb(0x60);
    count++;
    if(count%2==1)
            sound(al * 50);
    else nosound();
   oldhandler();
}
```

```
int main(void) {
    oldhandler =
            getvect(INTR);
    setvect(INTR, handler);
    while (count < 20);
    nosound();
    setvect(INTR, oldhandler);
   return 0;
}
```

Ví dụ

- Dùng chức năng 4Eh/4Fh của ngắt 21h viết các hàm FindFirst() và FindNext() để duyệt một thư mục
  - Tương tự hàm findfirst() và findnext() trong thư viện **dir.h**
- Hàm 4Eh
  - CX: chứa thuộc tính file
  - DS:DX trỏ tới chuôi ASCIIZ chưa tên file

  Trả về:
  - Cấu trúc thông tin file tại khôi DTA hiện tại
  - Cờ nhớ được đặt nếu có lỗi
- Hàm 4Fh: được gọi sau hàm 4Eh, không tham số
- Hàm 1Ah, đặt khối DTA (Disk transfer Area)hiện tại
  - DS:DX trỏ tới cấu trúc DTA mới

Ví dụ

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>

typedef struct{
    unsigned char Res[21];
    unsigned char Attr;
    unsigned int Time;
     unsigned int Date;
    unsigned long Size;
    char          Name[13];
}DIR_ENTRY;
```

```
int FindFirst(char path [],
    DIR_ENTRY * entry, int Attr){
    union REGS R;
    R.x.dx = FP_OFF(entry);
    R.h.ah = 0x1A;
    intdos(&R,&R);

    R.x.dx = FP_OFF(path);
    R.x.cx = Attr;
    R.h.ah = 0x4E;
    intdos(&R,&R);
    return R.x.cflag;
}
```

Ví dụ

```
int FindNext(){
    union REGS R;
    R.h.ah = 0x4F;
    intdos(&R,&R);
    return R.x.cflag;
}
```

```
int main(void){
   DIR_ENTRY Entry ;
   int done;
   printf("Directory listing of *.*\n");
   done = FindFirst("C:\\TC\\BIN\\*.exe",
                                &Entry,0xFFFF);
   while(!done){
      printf("%s \n", Entry.Name);
      done = FindNext();
   }
   return 0;
}
```