数组和指针类型的分配和访问

南京大学
计算机科学与技术系
袁春风
email: cfyuan@nju.edu.cn
2015.6

数组的分配和访问

- **数组元素在内存的存放和访问**
  - **例如，定义一个具有4个元素的静态存储型 short 数据类型数组A，可以写成“static short A[4];”**
  - **第 i（0≤i≤3）个元素的地址计算公式为&A[0]+2*i。**
  - **假定数组A的首地址存放在EDX中，i 存放在ECX中，现要将A[i]取到AX中，则所用的汇编指令是什么？**

movw (%edx, %ecx, 2), %ax **比例因子是2！**

其中，ECX为变址（索引）寄存器，在循环体中增量，

数组的分配和访问

- **填写下表**

数组定义	数组名	数组元素类型	数组元素大小（B）	数组大小（B）	起始地址	元素 i 的地址
char S[10]	S	char				
char * SA[10]	SA	char *				
double D[10]	D	double				
double * DA[10]	DA	double *				

数组的分配和访问

- **填写下表**

数组定义	数组名	数组元素类型	数组元素大小（B）	数组大小（B）	起始地址	元素 i 的地址
char S[10]	S	char	1	10	&S[0]	&S[0]+i
char * SA[10]	SA	char *	4	40	&SA[0]	&SA[0]+4*i
double D[10]	D	double	8	80	&D[0]	&D[0]+8*i
double * DA[10]	DA	double *	4	40	&DA[0]	&DA[0]+4*i

数组元素在内存的存放和访问

- 分配在静态区的数组的初始化和访问

```
int buf[2] = {10, 20};
int main ( )
{
    int i, sum=0;
    for (i=0; i<2; i++)
        sum+=buf[i];
    return sum;
}
```

buf是在静态区分配的数组，链接后，buf在可执行目标文件的数据段中分配了空间

```
08049080 <buf> :
08049080 :  0A 00 00 00 14 00 00 00
```

此时，buf=&buf[0]=0x08049080

编译器通常将其先存放到寄存器(如EDX)中

假定 i 被分配在ECX中，sum被分配在EAX中，则

“sum+=buf[i];”和 i++ 可用什么指令实现？

addl buf(, %ecx, 4), %eax 或 addl 0(%edx , %ecx, 4), %eax

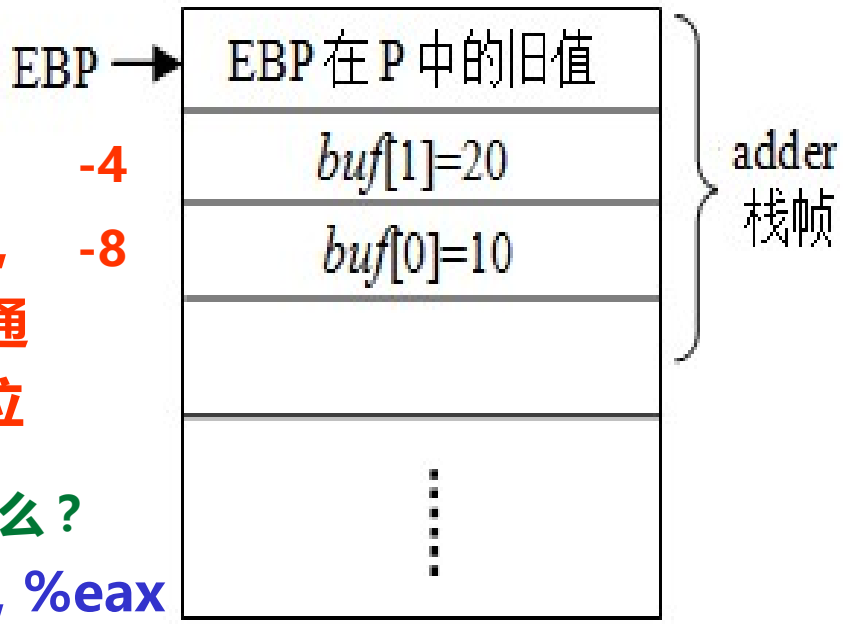
addl &1 , %ecx

数组元素在内存的存放和访问

- **auto型数组的初始化和访问**

```
int adder ( )
{
    int buf[2] = {10, 20};
    int i, sum=0;
    for (i=0; i<2; i++)
        sum+=buf[i];
    return sum;
}
```

分配在栈中，故数组首址通过EBP来定位



EDX、ECX各是什么？

```
addl (%edx, %ecx, 4), %eax
```

对buf进行初始化的指令是什么？

```
movl $10, -8(%ebp)    //buf[0]的地址为R[ebp]-8，将10赋给buf[0]
movl $20, -4(%ebp)    //buf[1]的地址为R[ebp]-4，将20赋给buf[1]
```

若buf首址在EDX中，则获得buf首址的对应指令是什么？

```
leal -8(%ebp), %edx  //buf[0]的地址为R[ebp]-8，将buf首址送EDX
```

数组元素在内存的存放和访问

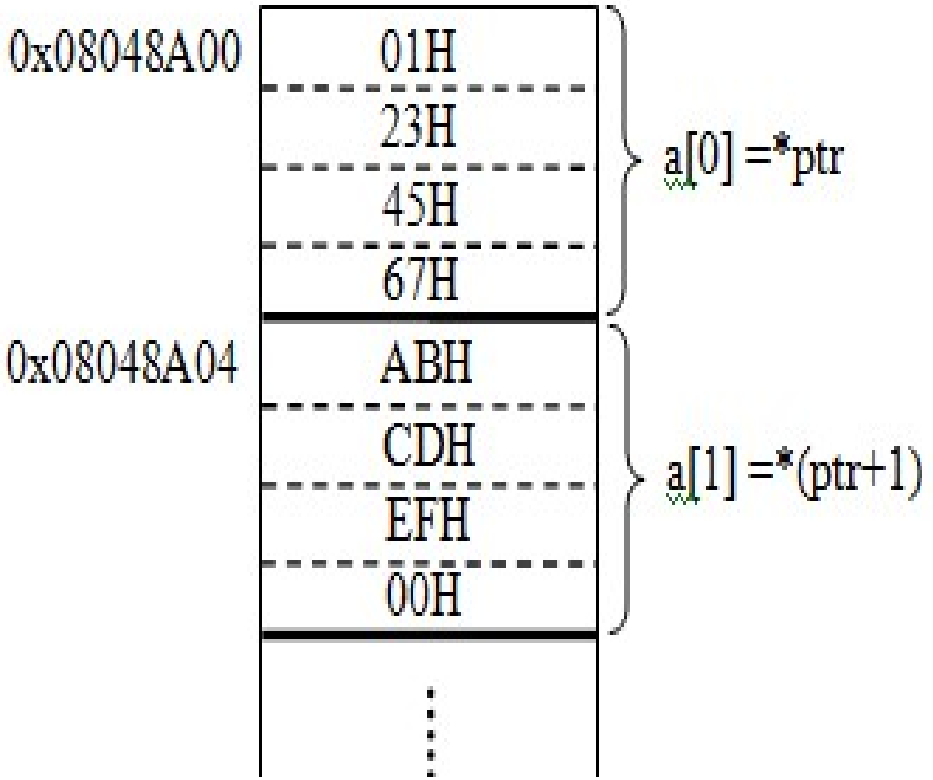
- **数组与指针**

✓ **在指针变量目标数据类型与数组类型相同的前提下，指针变量可以指向数组或数组中任意元素**

✓ **以下两个程序段功能完全相同，都是使ptr指向数组a的第0个元素a[0]。a的值就是其首地址，即a=&a[0]，因而a=ptr，从而有&a[i]=ptr+i=a+i以及a[i]=ptr[i]=*(ptr+i)=*(a+i)。**

（1）int a[10];
int *ptr=&a[0];

（2）int a[10], *ptr;
ptr=&a[0];



小端方式下a[0]=?,a[1]=?
a[0]=0x67452301, a[1]=0x0efcdab
数组首址0x8048A00在ptr中，ptr+i 并不是用0x8048A00加 i 得到，而是等于0x8048A00+4*i

数组元素在内存的存放和访问

- **数组与指针**

序号	表达式	类型	值的计算方式	汇编代码
1	A	int *		
2	A[0]	int		
3	A[i]	int		
4	&A[3]	int *		
5	&A[i]-A	int		
6	*(A+i)	int		
7	*(&A[0]+i-1)	int		
8	A+i	int *		

问题：

假定数组A的首址SA在ECX中，i在EDX中，表达式结果在EAX中，各表达式的计算方式以及汇编代码各是什么？

2、3、6和7对应汇编指令都需访存，指令中源操作数的寻址方式分别是“基址”、“基址加比例变址”、“基址加比例变址”和“基址加比例变址加位移”的方式，因为数组元素的类型为int型，故比例因子为4。

数组元素在内存的存放和访问

- **数组与指针** **假设A首址SA在ECX，i 在EDX，结果在EAX**

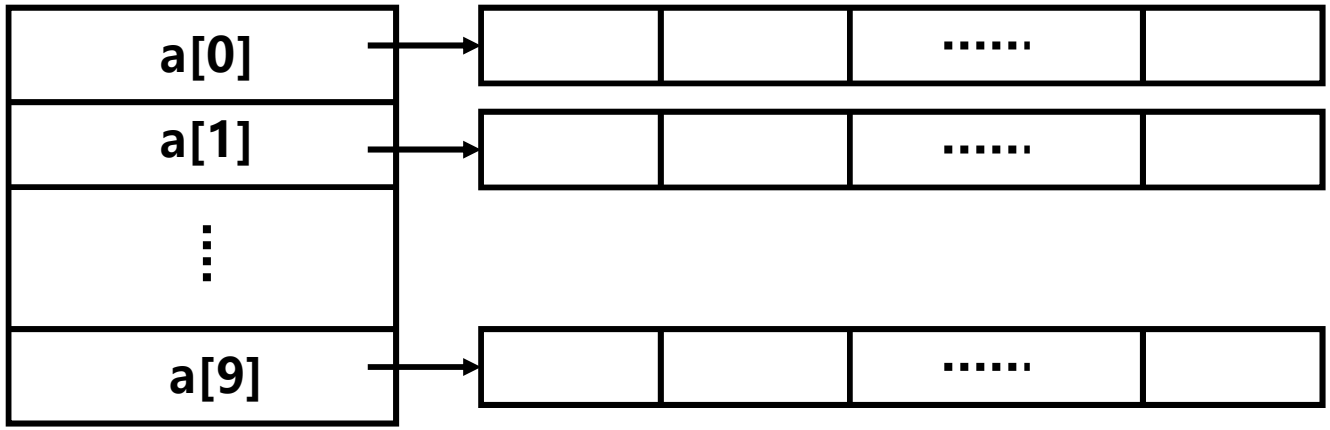
序号	表达式	类型	值的计算方式	汇编代码
1	A	int *	*SA*	leal (%ecx), %eax
2	A[0]	int	M[*SA*]	movl (%ecx), %eax
3	A[i]	int	M[*SA*+4**i*]	movl (%ecx, %edx, 4), %eax
4	&A[3]	int *	*SA*+12	leal 12(%ecx), %eax
5	&A[i]−A	int	(*SA*+4**i*−*SA*)/4=*i*	movl %edx, %eax
6	*(A+i)	int	M[*SA*+4**i*]	movl (%ecx, %edx, 4), %eax
7	*(&A[0]+i−1)	int	M[*SA*+4**i*−4]	movl −4(%ecx, edx, 4), %eax
8	A+i	int *	*SA*+4**i*	leal (%ecx, %edx, 4), %eax

2、3、6和7对应汇编指令都需访存，指令中源操作数的寻址方式分别是“基址”、“基址加比例变址”、“基址加比例变址”和“基址加比例变址加位移”的方式，因为数组元素的类型为int型，故比例因子为4 。

数组元素在内存的存放和访问

- **指针数组和多维数组**
  - **由若干指向同类目标的指针变量组成的数组称为指针数组。**
  - **其定义的一般形式如下：**

    存储类型 数据类型 *指针数组名[元素个数]；
  - **例如，“int *a[10];”定义了一个指针数组a，它有10个元素，每个元素都是一个指向int型数据的指针。**
    - **一个指针数组可以实现一个二维数组。**



数组元素在内存的存放和访问

- **指针数组和多维数组**　　按行优先方式存放数组元素
  - 计算一个两行四列整数矩阵中每一行数据的和。

```
main ( )
{
   static short num[ ][4]={ {2, 9, -1, 5},
                            {3, 8, 2, -6}};
   static short *pn[ ]={num[0], num[1]};
   static short s[2]={0, 0};
   int i, j;
   for (i=0; i<2; i++) {
      for (j=0; j<4; j++)
           s[i]+=*pn[i]++;
      printf (sum of line %d : %d\n” , i+1, s[i]);
   }
}
```

当i=1时，pn[i]=*(pn+i)=M[pn+4*i]=0x8049308

若处理“s[i]+=*pn[i]++;”时 i 在ECX，s[i]在AX，pn[i]在EDX，则对应指令序列可以是什么？

```
movl   pn(,%ecx,4), %edx
addw   (%edx), %ax
addl   $2, pn(, %ecx, 4)
```

pn[i]+“1”→pn[i]

若num=0x8049300,则num、pn和s在存储区中如何存放？

```
08049300 <num>:
08049300 :   02 00 09 00 ff ff 05 00 03 00 08 00 02 00 fa ff
08049310 <pn>:
08049310 :   00 93 04 08 08 93 04 08
08049318<s>:
08049318 :   00 00 00 00
```

num=num[0]=&num[0][0]=0x8049300

pn=&pn[0]=0x8049310
pn[0]=num[0]=0x8048300
pn[1]=num[1]=0x8048308