Тема 7

Работа с массивами

© 2011-2019

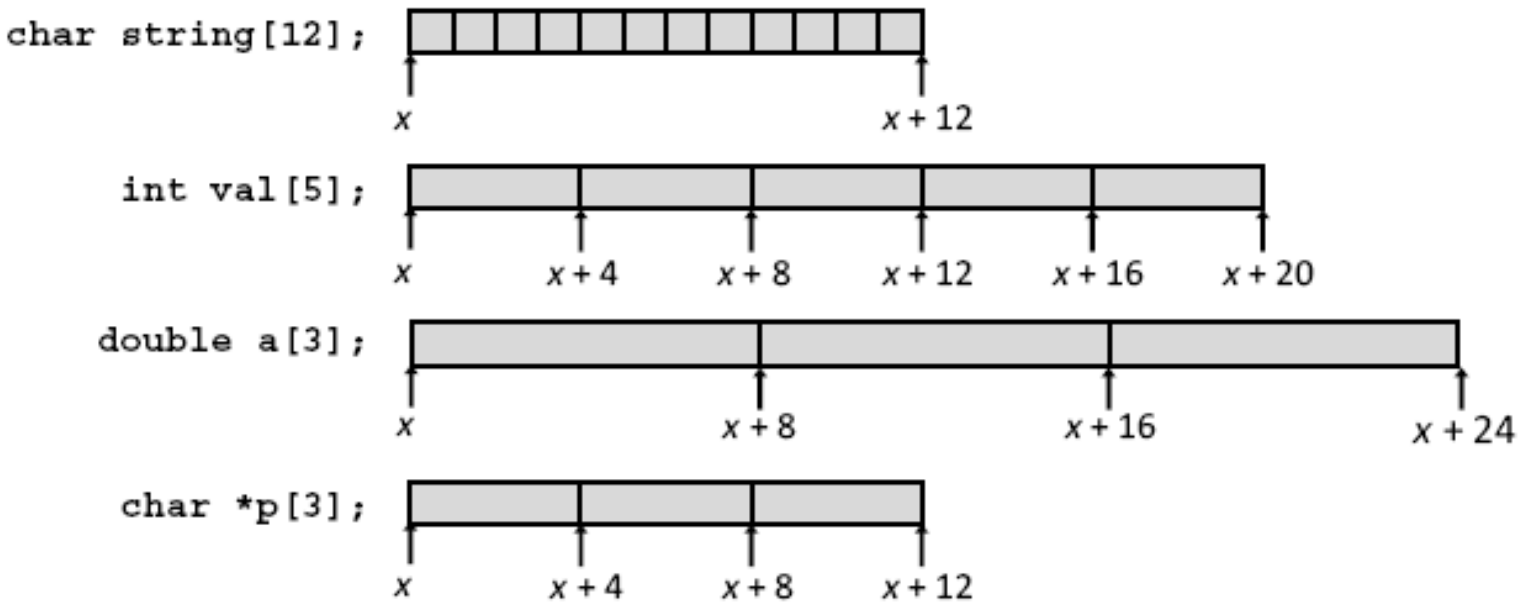
Работа с одномерными массивами

- Массивы – размещение в памяти

  `T A[L];`

  – Массив элементов типа T, размер массива – L

  – Массив располагается в непрерывном блоке памяти размером *L* * **`sizeof`** (*T*) байт

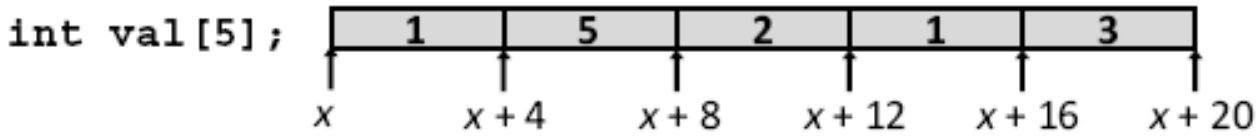


Работа с одномерными массивами

- Доступ к элементам массива

```
T   A[L];
```

  - Массив элементов типа T, размер массива – L
  - Идентификатор A может использоваться как указатель на элемент массива с индексом 0. Тип указателя – T*



- Ссылка — Тип — Значение

Ссылка	Тип	Значение
`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*

Работа с одномерными массивами

Для доступа к элементам массива с использованием механизма индексации необходимо хранить индекс в регистре общего назначения. При этом возможны варианты:

- в регистре хранится смещение в байтах относительно начала массива;
- в регистре хранится смещение в байтах относительно начала сегмента;
- в регистре хранится индекс, а смещение рассчитывается при обращении к массиву.

Для доступа к элементу массива допускается использовать два регистра.

Работа со статическими массивами (пример 1)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
    int A[] = {21, 4, 32, 45, 67, -21, 34, 56, 21, 3};
    int N = 10, rezt;
   _asm {
        mov   ecx, N
        xor   ebx, ebx     // смещение относительно
                           // начала массива
        xor   eax, eax     // для суммы
cycle:
        add   eax, A[ebx] //базовая адресация
        add   ebx, 4
        dec   ecx
        cmp   ecx, 0
        jne   cycle
        mov   rezt, eax
   }
   cout <<  rezt << "\n"; }
```

Работа со статическими массивами (пример 2)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
    int A[] = {21, 4, 32, 45, 67, -21, 34, 56, 21, 3};
   int N = 10, rezt;
   _asm {
       mov   ecx, N
       xor   esi, esi //индекс элемента массива
       xor   eax, eax
cycle:
       add   eax, A[4*esi]
       inc   esi
       dec   ecx
       cmp   ecx, 0
       jne   cycle
       mov   rezt, eax
   }
   cout <<  rezt << "\n";
}
```

Работа со статическими массивами (пример 3)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
  int A[] = {21, 4, 32, 45, 67, -21, 34, 56, 21, 3};
   int N = 10, rezt;
  _asm {
      mov   ecx, N
      lea   ebx, A       //смещение относительно
                         // начала сегмента
      xor   eax, eax
cycle:
      add   eax, [ebx]
      add   ebx, 4
      dec   ecx
      cmp   ecx, 0
      jne   cycle
      mov   rezt, eax
  }
  cout <<  rezt << "\n"; }
```

Работа со статическими массивами(пример 4)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
    int A[] = {21, 4, 32, 45, 67, -21, 34, 56, 21, 3};
    int N = 10, rezt;
    _asm {
      mov   ecx, N
      lea   ebx, A //смещение относительно начала сегмента
      xor   eax, eax
      xor   edi, edi  //смещение относительно
                      //начала массива
cycle:
      add   eax, [ebx][edi] //базово-индексная адресация
      add   edi, 4
      dec   ecx
      cmp   ecx, 0
      jne   cycle
      mov   rezt, eax
      }
      cout <<  rezt << "\n"; }
```

Работа с динамическими массивами (пример 5)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
    int *A = new int[10]; //заполнение массива A
    int N = 10, rezt;
    _asm {
       mov   ecx, N
       mov   ebx, A //смещение относительно начала сегмента
       xor   eax, eax
       xor   edi, edi    //смещение относительно начала
                         //массива
cycle:
       add   eax, [ebx][edi] //базово-индексная адресация
       add   edi, 4
       dec   ecx
       cmp   ecx, 0
       jne   cycle
       mov   rezt, eax
       }
     cout <<  rezt << "\n";}
```

Работа с динамическими массивами (пример 6)

Задача: найти сумму первых *N* элементов массива *A*

```
void main() {
   int *A = new int[10]; //заполнение массива A
   int N = 10, rezt;
   _asm {
       mov   ecx, N
       mov   ebx, A       //смещение относительно начала
                                 //сегмента
       xor   eax, eax     //сумма
cycle:
       add   eax, [ebx][4*ecx-4] //расчет справа налево
       loop  cycle
end_cycle:
       mov   rezt, eax
   }
   cout <<  rezt << "\n";
}
```

Работа с массивами (пример 7)

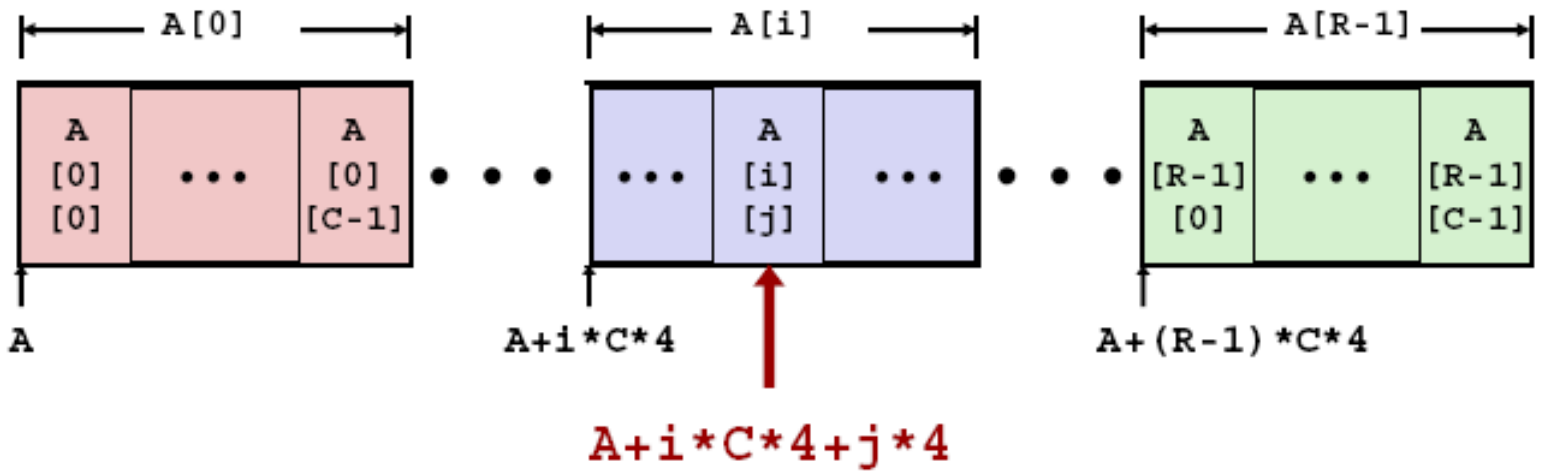
Задача: найти количество элементов беззнакового массива *A* из *N* чисел, стоящих перед первым нулем

```
void main() {
   unsigned A[10]; //заполнение массива A
    int N = 10, rezt;
   _asm {
            mov   ecx, N
            xor   ebx, ebx
            xor   eax, eax  // для количества
cycle:      cmp   A[ebx], 0
            je    m1
            inc   eax
            add   ebx, 4
m1:         loopne      cycle
            mov   rezt, eax
   }
   cout <<  rezt << "\n";}
```

Работа с двумерными массивами

- Элементы массива
  - **A[i][j]** элемент типа *T*, который требует *K* байт
  - Адрес элемента **A** + *i* * (*C* * *K*) + *j* * *K* = *A* + (*i* * *C* + *j*)* *K*

```
int A[R][C];
```



Работа с двумерными статическими массивами (пример)

Задача: сколько раз число встречается в матрице (базово-индексная адресация с масштабированием)

```
void main() {
    short array[5][2]=
                {{1,2},{3,4},{5,6},{7,3},{9,0}};
    short elem=3,    //элемент для поиска
          foundtime=0; //количество
  __asm
  {
     push esi
     xor  eax, eax
```

Работа с двумерными массивами (пример)

```
      lea ebx, array// ebx = строки в матрице
      mov ecx,5 // внешний цикл
external:       //внешний цикл по строкам
//сохранение в стеке счётчика внешнего цикла
     push ecx
  //число для внутреннего цикла (по столбцам)
     mov  ecx,2
     mov  esi,0// esi = столбцы в матрице
internal: //внутренний цикл по столбцам в строке
  // в ax 1-й элемент матрицы
     mov  ax, word ptr [ebx][esi*2]
```

Работа с двумерными массивами (пример)

```
 // сравниваем содержимое текущего элемента в ax
// с искомым элементом:
      cmp ax, elem
// если текущий не совпал с искомым,
// то переход на next для продолжения поиска
      jne next
// иначе увеличиваем счётчик совпавших
     inc  foundtime
next: // передвижение на следующий элемент в
  строке
      inc esi
      loop internal // цикл по строке ecx=2 раз
```

Работа с двумерными массивами (пример)

```
//продвижение в матрице
      pop ecx    // восстанавливаем ecx из
                 // стека(5-…)
      add ebx, 4  //передвигаемся на следующую
                    // строку
      loop external // цикл (внешний)
    pop esi
  }
  if (foundtime)
     cout << "Такой элемент в массиве
  присутствует " << foundtime << " раз " << endl;
  else cout << "Нет такого элемента в массиве!"
  << endl;
}
```

Решение задачи с динамическими массивами

```
void main() {
  int n;                  //Размерность матрицы
  short ** matrix;   //Указатель на вершину
  матрицы
  short elem=3,                  //элемент для поиска
        foundtime=0;       //количество
  cin >> n;
     //Создание матрицы
  matrix = new short* [n];
  for (int i=0; i<n; i++)
     matrix[i] = new short [n];
  ... // заполнение
```

Работа с двумерными динамическими массивами

```
    __asm
{
    push esi
    xor  eax, eax
    xor  edx, edx
    mov  ebx, matrix     // ebx = адрес матрицы
    mov  ecx, n      // число для внешнего цикла
external:            //внешний цикл по строкам
    mov  edx,[ebx] // адрес строки в матрице
    push ecx
    mov  ecx,n
    xor  esi,esi    //Итератор столбца
```

Работа с двумерными динамическими массивами

```
internal: //внутренний цикл по столбцам в
          // строке
     mov  ax, word ptr [edx][esi*2]
     cmp  ax, elem
     jne  next
     inc  foundtime
next: // передвижение на следующий элемент в
     //строке
     inc  esi
     loop internal   // цикл по строке
```

Работа с двумерными динамическими массивами

```
    pop  ecx          // восстанавливаем ecx из
                      //стека(5-…)
    add  ebx, 4       // на следующую строку
    loop external // цикл (внешний)
    pop  esi
  }
  if (foundtime)
     cout << "Такой элемент в массиве
присутствует " << foundtime << " раз " << endl;
  else cout << "Нет такого элемента в массиве!"
  << endl;
}
```