# **Тема 7. Способы адресации и обработка массивов. Способы адресации.**КОП РР,РР Регистровая адресация КОП РР,ОП Прямая адресация КОП ОП,РР Косвенная адресация КОП РР,НД Непосредственная адрес-я КОП ОП,НД **7.1.Регистровая адресация** Операнды могут располагаться в любых регистрах общего назначения и сегментных регистрах. Для этого в тексте программы указывается название соответствующих регистров: mov ах, bх mov ecх, ebх mov аh, bl mov eах, bх //ошибка mov ecх, bх //ошибка mov аx, bl //ошибка

# **7.2. Непосредственная адресация (РД и ПД)** Некоторые команды позволяют указывать один из операндов непосредствен-но в тексте программы. Например: mov d,100 //помещает в d число 100 mov ecх, 100 //помещает в ecх 100. При такой адресации один операнд находится в регистре или в памяти, а второй записан в самой команде, причем ДЛИНА этих данных тоже ВСЕГДА ОДИНАКОВА!

# В качестве НЕПОСРЕДСТВЕННОГО ДАННОГО может выступать десятичное число, шестнадцатеричное число, символ и двоичное число. Например, можно записать такие команду mov eax,100h //помещает в eах число 256. mov d,100b //помещает в d число 4 mov cx, 1 // помещает в cx число 1. mov dl,’F’ **7.3. Прямая адресация** Если у операнда, располагающегося в памяти, известен адрес, то его можно использовать. Пусть а статический массив: int a[]={1,2,5,4}; Можно записать следующие команды: mov eax,a //1 mov ecx,a+4 //2 mov ebx, a+8 //5 mov edx, a+12 //4

# int a[]={1,2,5,4}; 002643FF mov dword ptr [a],1 00264406 mov dword ptr [ebp-38h],2 0026440D mov dword ptr [ebp-34h],5 00264414 mov dword ptr [ebp-30h],4 \_\_asm { mov eax,a //1 0026441B mov eax,dword ptr [a] mov ecx,[a+4] //2 0026441E mov ecx,dword ptr [ebp-38h] mov ebx,a+8 //5 00264421 mov ebx,dword ptr [ebp-34h] mov edx, a+12 //4 00264424 mov edx,dword ptr [ebp-30h] }

# ecx 2 unsigned int ebx 5 unsigned int edx 1 unsigned int esi 8000118 unsigned int eax 1 unsigned int void main() //прямая адресация плохо!!! { int i, n=3,c; int \*a=new int [n]; //динамический for (i=0;i<n;i++) cin>> a[i]; //10 20 30 \_\_asm { mov ecx,a mov ebx,a+4 mov edx,[a+4] mov esi, a mov eax, [esi] mov c,eax } }

# Результат выполнения ecx 23105440 unsigned int ebx 3435973836 unsigned int edx 3435973836 unsigned int esi 23105440 unsigned int eax 10 unsigned int \_\_asm { mov ecx,a 0009446B mov ecx,dword ptr [a] mov ebx,a+4 0009446E mov ebx,dword ptr [ebp-28h] mov edx,[a+4] 00094471 mov edx,dword ptr [ebp-28h] mov esi, a 00094474 mov esi,dword ptr [a] mov eax, [esi] 00094477 mov eax,dword ptr [esi] mov c,eax 00094479 mov dword ptr [c],eax }

# Пример 7.1. Сумма элементов массива int a[]={1,2,3}; int c; \_\_asm { mov esi, 0 mov eax, a add eax, a+4 add eax, a+8 mov c,eax } *//результат c=6*

# Пример 7.2. Сумма элементов массива \_int16 a[]={1,2,3,4}; int c; \_\_asm { mov esi, 0 mov eax, a add eax, a+2 add eax, a+4 add eax, a+6 mov c,eax } *//результат c=10*

# 7.4. **Косвенная адресация** По аналогии с регистровыми и непосредственными операндами адрес операнда в памяти также можно не указывать, а хранить в любом регистре. Адрес операнда можно считывать из ЕАХ, ЕВХ, ЕСХ, EDX, ESI, EDI, ЕВР и ESP,ВХ, SI, DI ,ВР.

# Например, следующая команда помещает в регистр EАХ слово из ячейки памяти, адрес сегмента которой находится в DS а смещение - в EВХ: mov eах,[ebх] Косвенная адресация включает в себя следующие   индексную   базовую   базово-индексную

# 7.5. Индексная адресация При задании адреса используется один из индексных регистров esi ( si ) или edi ( di ). При таком способе адресации адрес области памяти может иметь вид :       индексный регистр в квадратных скобках. Например, команда вида mov [esi], 10 //адрес ->esi

#       некоторое символическое имя и индексный регистр в квадратных скобках. mov arr[esi],10 // число ->esi       в квадратных скобках индексный регистр, символическое имя константы, соединенные знаком +. mov [arr+ esi],10 mov [arr+ esi+2],10

# Пример 7.3. Сумма элементов массива int arr[]={1,2,3}; int c; \_\_asm { mov esi, 0 mov eax, arr[esi] add esi,4 add eax, arr[esi] add esi,4 add eax, arr[esi] mov c,eax }*//результат c=6*

# Пример 7.4. Сумма элементов массива int arr[]={1,2,3}; int c; \_\_asm { mov esi, 0 mov eax, arr[esi] add esi,4 add eax, [arr+esi] add eax, [arr+esi+4] mov c,eax } *//результат c=6*

# Пример 7.5. // Применение индексной адресации char slovo1[]="HELLO"; char slovo2[]="HELLO"; char slovo3[]="HELLO"; int c; \_\_asm { mov esi,2 mov slovo1[2],'\0‘ *//результат* slovo1*= HE* mov slovo2[esi+1],'\0' mov slovo3[esi+2],'\0‘ //slovo2*= HEL* slovo3*= HELL* }

# Пример 7.6. Сумма элементов массива int arr[]={1,2,7,4,2,5,6}; int c; \_\_asm { mov ecx,7 mov eax,0 mov esi, 0 for\_: add eax, arr[esi] add esi,4 loop for\_ mov c,eax }

# Пример 7.7. Сумма элементов массива int \*b, int n,c; cin>>n; a=new int [n]; for (i=0;i<n;i++) cin>> \*b++; \_\_asm { mov ecx,n mov eax,0 mov esi, b for\_: mov eax, [esi] add esi,4 loop for\_ mov c,eax }

# **7.6. Базовая адресация. Если применяется базовая адресация**, то при задании адреса используется один из базовых регистров : либо ЕBX(ВХ) либо BP При таком способе адресации адрес области памяти, куда помещается информация, может быть записан :       заключением в квадратные скобки базового регистра : MOV [ebx],10       заключением в квадратные скобки не только базового регистра, но и некоторого символического имени: MOV [x+ebx],10       заключением в квадратные скобки не только базового регистра и некоторого символического имени, но и некоторого числа, называемого смещением: MOV [x+ebx+4],10       Допустимые формы записи этого способа адресации:

# Пример 7.8. // сумма двух чисел #include <stdio.h> #include <stdlib.h> void main() { int a=3,\*b,c; b=new int; \*b=4; \_asm { mov eax, a mov ebx,b add eax,[ebx] mov c, eax } cout<<c; //Результат 3+4=7 } Пример 7.9. // // сумма двух чисел void main() { int c,\*b; b=new int [2]; \*b=4; \*(b+1)=3; \_asm { mov eax,0 mov ebx,b add eax,[ebx] add eax ,[ebx+4] // add eax ,4[ebx] // add eax ,[ebx]+4 mov c, eax} cout<<c; }

# Пример 7.10. Сумма элементов массива int a[]={1,2,7,4,2}; int c; \_\_asm { mov ecx,5 mov eax,0 mov ebx, 0 for\_: add eax, a[ebx] //?????? add ebx,4 loop for\_ mov c,eax }

# **7.7. Адресация по базе с индексированием (базово—индексная)** *Базово—индексная* адресация , это когда при задании адреса используются один из базовых регистров : либо ЕBX(BX) либо ЕBP ( BР) и один из индексных регистров либо ЕSI (SI) либо ЕDI (DI) При таком способе адресации адрес области памяти, куда заносится информация, может быть записан int A[100] или int \*А:

#      либо заключением в квадратные скобки базового и индексного регистров, например : MOV[EBX,ESI],25      либо заключением в квадратные скобки не только базового и индексного регистров, но и некоторого символического имени, например MOV А[ЕBX,ЕSI],2020      либо заключением в квадратные скобки не только базового регистра и индексного регистра, некоторого символического имени, но и некоторого числа, называемого смещением. Например, MOV [A+ЕBX+ЕSI+4],2020

# В этом методе адресации смещение операнда в памяти вычисляется как сумма чисел, содержащихся в двух регистрах, и смещения, если оно указано. Все эти команды представляют собой разные формы записи одного и того же действия: mov eax,[ebx+esi+4] mov eax,[ebx][esi]+4 mov eax,[ebx+4][esi] mov eeax,[ebx][esi+4] mov eax,4[ebx][esi] Если int \*A => mov ebx,A и число в esi int A[100] => lea ebx,A и число в esi

# Пример 7.11. // сумма двух чисел void main() {int c,\*b; b=new int[2]; \*b=4; \*(b+1)=3; asm { mov eax,0 mov ebx,b mov esi,0 add eax,[ebx] add esi,4 add eax,[ebx+esi] //add eax,[ebx][esi] mov c, eax} **7.8. Адресация по базе с индексированием и масштабированием** Это самая полная схема адресации, в которую входят все случаи, рассмотренные ранее как частные. Смещение может быть байтом или двойным словом. Если ESP или ЕВР используются в роли базового регистра, селектор сегмента операнда берется по умолчанию из регистра SS, во всех остальных случаях - из DS.

# Пример 7.12 // обмен в массиве двух элементов 1 и 2 void main() { int b[]={4,3,2}; { mov eax, b[esi] add esi, 4 mov edx,b[esi] mov b[esi],eax mov b[esi-4],edx } }

# void main() {int \*b; b=new int[3]; \*b=4; \*(b+1)=3; \*(b+2)=2; \_\_asm { mov ebx, b //адрес b mov esi, 4 mov eax, [ebx][esi] //eax=3 add esi, 4 mov edx,[ebx][esi] //edx=2 mov [ebx][esi],eax mov [ebx-4][esi],edx // sub esi, 4 // mov [ebx][esi],edx } }

# void main() {int \*b; b=new int[3]; \*b=4; \*(b+1)=3; \*(b+2)=2; \_\_asm { mov ebx, b //адрес b mov esi, 1 mov eax, [ebx][esi\*4]// eax=3 add esi, 1 mov edx,[ebx][esi\*4] //edx=2 mov [ebx][esi\*4],eax mov [ebx-4][esi\*4],edx // sub esi, 1 // mov [ebx][esi\*4],edx } }

# Пример 7.13 Поиск максимального элемента int \*A, n, Amax; cin>>n; A = new int [n];  //ввод чисел массива \_asm { mov ecx,n dec ecx mov esi,A // адрес массива mov eax,[esi] // элемент, адрес которого в esi for1: add esi,4 cmp eax,[esi] cmovl eax,[esi] // переслать, если меньше loop for1 mov Amax,eax }

# Пример 7.14 Поиск максимального элемента int A[100], n, Amax; cin>>n;  \_asm { mov ecx,n dec ecx xor esi,esi mov eax,A[esi] // первый элемент массива for1: add esi,4 cmp eax,A[esi] cmovl eax,A[esi] // переслать, если меньше loop for1 mov Amax,eax }

# Пример 7.15 Сортировка элементов массива void sort (int \*ar, int n ) //void sort (int ar[], int n ) { */\*for ( int i=0 ; i<n; i++) for( int j=0; j<n-i-1 ; j++) if (ar[j]>ar[j+1]) swap(ar[j],ar[j+1]);\*/* \_asm { mov ecx,n dec ecx

# loop1: push ecx mov esi,ar loop2: mov eax,[esi] cmp eax,[esi+4] jle then\_ xchg eax,[esi+4] mov [esi],eax then\_: add esi,4 loop loop2 pop ecx loop loop1 }

# void main() { int i, n; cin>>n; // int a[100]; int \*a=new int[n]; // ввод чисел или получение случайных sort(a,n); for (i=0;i<n;i++) cout<< a[i]<<" "; }