# Лабораторная работа №1. Регистры, данные и команды пересылки данных

**Цель работы**

Изучение регистров общего назначения (РОН) и команд пересылки данных.

**Постановка задачи**

1. Занести число из столбца «Число 1» таблицы 1 и строки, соответствующей заданному варианту, в любой РОН;
2. Занести число из столбца «Число 2» таблицы 1 и строки, соответствующей заданному варианту, в незанятый РОН;
3. Занести число из столбца «Число 3» таблицы 1 и строки, соответствующей заданному варианту, в незанятый РОН;
4. Занести число из столбца «Число 4» таблицы 1 и строки, соответствующей заданному варианту, в незанятый РОН;
5. Занести число из столбца «Число 5» таблицы 1 и строки, соответствующей заданному варианту, в незанятый РОН;
6. Занести число из столбца «Число 6» таблицы 1 и строки, соответствующей заданному варианту, в незанятый РОН;
7. Обменять числа, хранящиеся в РОН после выполнения пунктов 1 и 2, между собой, 4-мя разными способами, не потеряв информацию в занятых РОН;
8. Обменять числа, хранящиеся в РОН после выполнения пунктов 3 и 4, между собой, 4-мя разными способами, не потеряв информацию в занятых РОН;
9. Обменять числа, хранящиеся в РОН после выполнения пунктов 5 и 6, между собой, 4-мя разными способами, не потеряв информацию в занятых РОН;
10. Переслать числа, оказавшиеся в РОН после выполнения пункта с номером, взятым из столбца «Пункт 1» таблицы 1 и строки, соответствующей заданному варианту, в любые 16-разрядные РОН без потери знака;
11. Переслать числа, оказавшиеся в РОН после выполнения пункта с номером, взятым из столбца «Пункт 2» таблицы 1 и строки, соответствующей заданному варианту, в любые 32-разрядные РОН расширенные нулем;

***Примечание: Регистр ESP и EBP использовать НЕЛЬЗЯ !!!***

**Содержание отчета**

Отчет по лабораторной работе должен содержать номер, название, цель лабораторной работы, задание на лабораторную работу, текст программы на языке ассемблера, распечатки формы в активном состоянии с результатами выполнения лабораторной работы в виде дампа памяти и фрагмента программы, содержащей ассемблерную вставку, а также вывод по работе.

**Варианты заданий**

Таблица 1.

| Номер варианта | Число 1 | Число 2 | Число 3 | Число 4 | Число 5 | Число 6 | Пункт 1 | Пункт 2 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 3 050 316 749 | 857 033 805 | 64 001 | 27 946 | 134 | 47 | 8 | 8 |
| 2 | 3 933 384 589 | 971 413 658 | 48 330 | 22 467 | 162 | 5 | 8 | 9 |
| 3 | 3 968 432 562 | 250 761 973 | 45 938 | 25 727 | 210 | 42 | 9 | 8 |
| 4 | 2 669 592 357 | 6 213 853 | 47 036 | 16 800 | 145 | 94 | 9 | 9 |
| 5 | 3 609 857 455 | 2 001 729 633 | 54 633 | 8 498 | 183 | 114 | 8 | 8 |
| 6 | 3 257 266 905 | 1 245 818 205 | 56 002 | 25 418 | 212 | 75 | 8 | 9 |
| 7 | 2 916 235 308 | 987 999 567 | 55 633 | 32 029 | 139 | 26 | 9 | 8 |
| 8 | 2 756 406 257 | 1 506 812 197 | 63 484 | 8 425 | 153 | 4 | 9 | 9 |
| 9 | 4 034 305 306 | 392 802 862 | 35 094 | 11 811 | 137 | 7 | 8 | 8 |
| 10 | 3 159 448 700 | 1 544 211 091 | 57 236 | 22 027 | 243 | 108 | 8 | 9 |
| 11 | 2 488 047 062 | 2 028 847 670 | 47 579 | 22 442 | 250 | 66 | 9 | 8 |
| 12 | 4 247 776 274 | 103 373 407 | 46 130 | 29 961 | 151 | 56 | 9 | 9 |
| 13 | 3 552 292 199 | 1 628 905 995 | 58 404 | 13 389 | 147 | 62 | 8 | 8 |
| 14 | 4 008 698 874 | 306 001 926 | 52 347 | 20 501 | 153 | 74 | 8 | 9 |
| 15 | 2 433 934 443 | 2 015 149 002 | 56 190 | 5 075 | 236 | 69 | 9 | 8 |
| 16 | 3 243 508 463 | 1 098 303 011 | 61 752 | 25 291 | 227 | 60 | 9 | 9 |
| 17 | 3 598 586 003 | 1 700 879 456 | 51 395 | 12 158 | 150 | 72 | 8 | 8 |
| 18 | 3 948 008 498 | 1 823 230 391 | 56 897 | 22 903 | 175 | 89 | 8 | 9 |
| 19 | 4 048 479 298 | 1 670 322 348 | 45 257 | 25 552 | 233 | 110 | 9 | 8 |
| 20 | 2 999 718 865 | 2 089 409 762 | 59 534 | 26 378 | 203 | 59 | 9 | 9 |
| 21 | 2 711 728 559 | 1 472 656 564 | 62 574 | 32 346 | 204 | 51 | 8 | 8 |
| 22 | 3 683 606 123 | 335 322 861 | 64 260 | 6 699 | 224 | 112 | 8 | 9 |
| 23 | 3 645 284 490 | 1 559 350 478 | 59 482 | 3 327 | 149 | 78 | 9 | 8 |
| 24 | 3 821 456 732 | 183 754 094 | 42 930 | 5 063 | 223 | 66 | 9 | 9 |
| 25 | 2 516 094 474 | 1 472 212 419 | 51 469 | 16 564 | 145 | 86 | 8 | 8 |
| 26 | 4 194 231 035 | 954 481 758 | 33 811 | 5 512 | 171 | 88 | 8 | 9 |
| 27 | 2 398 029 387 | 3 261 100 | 63 166 | 13 971 | 224 | 97 | 9 | 8 |
| 28 | 3 401 178 541 | 2 021 332 153 | 38 118 | 32 100 | 184 | 55 | 9 | 9 |
| 29 | 3 312 549 825 | 653 774 000 | 45 597 | 12 799 | 246 | 77 | 8 | 8 |
| 30 | 4 130 192 976 | 1 564 933 669 | 55 114 | 16 506 | 175 | 83 | 8 | 9 |
| 31 | 4 138 113 708 | 1 815 058 639 | 42 042 | 5 573 | 219 | 58 | 9 | 8 |
| 32 | 4 286 093 495 | 1 084 073 624 | 55 016 | 31 876 | 227 | 110 | 9 | 9 |
| 33 | 3 121 711 706 | 1 966 382 217 | 38 040 | 1 449 | 242 | 90 | 8 | 8 |
| 34 | 2 374 644 833 | 962 127 780 | 54 116 | 25 875 | 199 | 88 | 8 | 9 |
| 35 | 2 809 635 978 | 1 981 105 341 | 34 106 | 5 212 | 189 | 61 | 9 | 8 |
| 36 | 4 138 500 438 | 1 767 390 115 | 39 427 | 21 800 | 211 | 18 | 9 | 9 |
| 37 | 2 156 758 467 | 660 985 841 | 35 423 | 28 994 | 176 | 22 | 8 | 8 |
| 38 | 4 217 006 650 | 1 168 671 945 | 62 887 | 6 141 | 164 | 80 | 8 | 9 |
| 39 | 4 047 827 015 | 1 821 383 074 | 58 379 | 22 630 | 191 | 96 | 9 | 8 |
| 40 | 3 559 739 825 | 256 920 863 | 33 198 | 6 043 | 165 | 66 | 9 | 9 |
| 41 | 3 780 056 098 | 1 603 982 851 | 51 278 | 17 711 | 183 | 62 | 8 | 8 |
| 42 | 3 797 296 420 | 1 908 629 559 | 48 250 | 21 347 | 240 | 120 | 8 | 9 |
| 43 | 3 736 924 978 | 1 296 290 749 | 32 944 | 32 763 | 157 | 44 | 9 | 8 |
| 44 | 2 543 238 445 | 372 228 216 | 44 181 | 4 117 | 131 | 106 | 9 | 9 |
| 45 | 2 786 920 617 | 47 245 825 | 35 912 | 17 475 | 240 | 82 | 8 | 8 |
| 46 | 2 570 192 319 | 632 432 585 | 41 490 | 8 458 | 202 | 120 | 8 | 9 |
| 47 | 3 536 602 848 | 621 455 134 | 33 441 | 28 950 | 193 | 116 | 9 | 8 |
| 48 | 2 518 445 482 | 93 653 929 | 50 079 | 28 225 | 153 | 38 | 9 | 9 |
| 49 | 3 540 914 521 | 1 864 588 387 | 43 010 | 15 728 | 200 | 28 | 8 | 8 |
| 50 | 3 553 032 756 | 2 081 070 464 | 40 595 | 567 | 161 | 16 | 8 | 9 |
| 51 | 3 234 044 570 | 1 430 006 929 | 49 279 | 11 831 | 209 | 34 | 9 | 8 |
| 52 | 3 161 648 983 | 1 268 418 336 | 60 595 | 2 650 | 149 | 27 | 9 | 9 |
| 53 | 3 322 204 893 | 256 698 229 | 49 297 | 8 110 | 253 | 5 | 8 | 8 |
| 54 | 4 161 566 815 | 1 396 434 740 | 37 590 | 17 767 | 241 | 49 | 8 | 9 |
| 55 | 4 219 198 386 | 769 555 691 | 45 524 | 28 107 | 183 | 100 | 9 | 8 |
| 56 | 3 057 003 449 | 2 122 839 732 | 62 337 | 5 848 | 167 | 86 | 9 | 9 |
| 57 | 3 222 886 638 | 50 497 194 | 55 352 | 26 187 | 141 | 28 | 8 | 8 |
| 58 | 4 075 678 955 | 1 348 534 018 | 37 269 | 1 168 | 236 | 70 | 8 | 9 |
| 59 | 2 417 251 151 | 1 059 178 430 | 65 004 | 29 622 | 241 | 102 | 9 | 8 |
| 60 | 2 782 443 731 | 1 904 828 683 | 65 166 | 20 558 | 228 | 37 | 9 | 9 |

**Краткие теоретические сведения**

Микропроцессор Intel семейства P6 содержит восемь регистров общего назначения (РОН), которые используются для хранения операндов и результатов команд, а также для формирования адреса при обращении к данным в памяти.

РОН микропроцессора Intel P6, их разрядность и вложенность приведены в таблице 2.

Таблица 2.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 | 16 | 15 |  | 8 | 7 |  | 0 | ← номера битов |
| EAX |  | | AX | | | | | |  |
|  | AH |  |  | AL |  |  |
| EBX |  | | BX | | | | | |  |
|  | BH |  |  | BL |  |  |
| ECX |  | | CX | | | | | |  |
|  | CH |  |  | CL |  |  |
| EDX |  | | DX | | | | | |  |
|  | DH |  |  | DL |  |  |
| EBP |  | | BP | | | | | |  |
|  |
| ESI |  | | SI | | | | | |  |
|  |
| EDI |  | | DI | | | | | |  |
|  |
| ESP |  | | SP | | | | | |  |
|  |

При обращении к ячейкам памяти, адрес ячейки памяти может указываться косвенно, т.е. адрес может быть указан в каком-либо РОН или с помощью математического выражения следующего вида:

Первое слагаемое называется базой, второе – индекс, умноженный на масштаб, третье – константа. Формула может состоять из любого сочетания слагаемых в квадратных скобках (напомним, что в квадратных скобках здесь указаны слагаемые, которые могут полностью отсутствовать) и любого элемента из каждой строчки в круглых скобках. Следует напомнить, что операнд, задающий адрес ячейки памяти, указывается в квадратных скобках.

Все команды на языке ассемблера записываются в виде мнемоники команды и разного количества операндов. Мнемоника это сокращение английских слов, выражающих смысл команды.

К командам пересылки данных относятся:

MOV - от англ. Move – движение, т.е. перемещение информации;

PUSH - от англ. Push – поместить, т.е. занести информацию;

POP - от англ. POP - выталкивание, т.е. забрать информацию;

XCHG - от англ. Exchange – обмен информацией;

BSWAP - от англ. Byte Swap – байтовый обмен;

MOVSX - от англ. Move with Sign Extended – перемещение информации со знаковым расширением;

MOVZX - от англ. Move with Zero Extended – перемещение информации с расширением нулем;

LEA - от англ. Load Effective Address – загрузить эффективный адрес.

**Команда MOV**

Формат команды: MOVП,И

где П – приемник информации. В качестве приемника может использоваться РОН или ячейка памяти;

И – источник информации. В качестве источника информации могут использоваться РОН, ячейка памяти или константа (непосредственные данные, указанные в самой команде ассемблера).

Команда MOV копирует данные из источника в приемник. Приемник и источник должны быть одинаковой размерности! Нельзя в одной команде MOV пересылать данные между двумя ячейками памяти! При записи в память, т.е., если в качестве приемника указан адрес ячейки памяти, информация размером в Byte (8 бит) будет записана непосредственно в ячейку памяти с адресом, указанным в команде. При записи информации размером в Word (2 байта или 16 бит), будут задействованы две соседние ячейки памяти, начиная с адреса, указанного в команде, причем первым будет записан младший байт информации, а в соседнюю ячейку (в сторону увеличения адресов), старший байт информации. При записи информации размером Integer (2 слова, 4 байта, 32 бита), будут задействованы четыре соседних ячейки памяти начиная с адреса, указанного в команде, причем, первым будет записан младший байт младшего слова информации, а в соседние ячейки (в сторону увеличения адресов), соответственно, старший байт младшего слова, младший байт старшего слова и старший байт старшего слова (младшим байтом или словом называют байт или слово с меньшими номерами битов, например, в слове младшим байтом называют байт, который состоит из битов с нулевого по седьмой, а старшим тот, который состоит из битов с восьмого по пятнадцатый).

Примеры:

MOV AL,10 Занести число 10 в регистр AL. После выполнения команды AL=0A.

MOV BX,$FA Занести число FA16 (25010) в регистр BX. В данной строке примера использован символ «$» перед числом, который является обязательным префиксом для шестнадцатеричного числа. После выполнения команды BX=00FA. Несмотря на то, что в команде указана байтовая константа, компилятор оценит приемник, в качестве которого здесь выступает 16-ти разрядный регистр, и автоматически расширит константу незначащими нулями слева.

MOV ECX,EDX Занести значение регистра EDX в регистр ECX. Совершенно очевидно, что EDX сохранит свое значение.

MOV [EBX],BH Занести в ячейку памяти (т.к. приемник указан в квадратных скобках) с адресом, хранящимся в регистре EBX, один байт (так как источником является 8-ми разрядный регистр) со значением, хранящимся в регистре BH.

MOV Byte Ptr [ESI], $F0 Занести в ячейку памяти (т.к. приемник указан в квадратных скобках) с адресом, хранящимся в регистре ESI, один байт со значением F016 (24010). В данном случае определить размер источника без дополнительных указаний не представляется возможным, так как это может быть и F016, и 00F016, и 000000F016 и любая из этих констант может быть записана в память, заняв, соответственно, одну, две или четыре соседних ячейки памяти. В таких случаях размер данных указывается специальным префиксом, в данном случае Byte Ptr, который четко указывает, что заноситься будет один байт.

MOV Word Ptr [ECX+EDX\*2], $1234 Занести в ячейку памяти с адресом, вычисленным по выражению ECX+EDX\*2, одно слово со значением 123416 (466010). При этом по адресу ECX+EDX\*2 будет занесен байт $34, а по адресу ECX+EDX\*2+1 байт $12.

MOV DWord Ptr [ECX+EBX+$A], $12345678 Занести в ячейку памяти с адресом, вычисленным по выражению ECX+EBX+$A, двойное слово со значением 1234567816 (1908874410). При этом по адресу ECX+EBX+$A будет занесен байт $78, по адресу ECX+EBX+$A+1 байт $56, по адресу ECX+EBX+$A+2 байт $34, а по адресу ECX+EBX+$A+3 байт $12.

**Команда PUSH**

Формат команды: PUSHИ

где И – источник информации. В качестве источника информации могут использоваться РОН, ячейка памяти или константа (непосредственные данные, указанные в самой команде ассемблера).

Команда PUSH копирует данные из источника в вершину стека. В качестве источника могут использоваться только 16-ти и 32-ух разрядные данные! Стеком называется область памяти в общем адресном пространстве, которая работает по принципу FILO (First In Last Out – первым вошел, последним вышел). Для работы со стеком используются специальные команды, а также стек используется автоматически при вызове подпрограмм. В самом простом виде, стек используется для быстрого временного сохранения и восстановления значений РОН. Языки высокого уровня используют стек для передачи параметров в процедуры и функции. В процессорах Intel P6 адрес вершины стека после занесения данных уменьшается.

Примеры:

PUSH AX Поместить в стек значение регистра AX.

PUSH Word Ptr $FA Поместить в стек значение слова FA16 (25010). Таким образом в стек запишется 00FA16.

PUSH DWord Ptr [EBX] Поместить в стек длинное слово из четырех соседних ячеек, первая из которых имеет адрес с номером, хранящимся в EBX.

Команда PUSH имеет модификации:

PUSHA – занести в стек все 16-ти разрядные РОН в следующем порядке AX,CX,DX,BX,SP,BP,SI,DI;

PUSHAD – занести в стек все 32-ух разрядные РОН в следующем порядке EAX,ECX,EDX,EBX,ESP,EBP,ESI,EDI;

**Команда POP**

Формат команды: POPП

где П – приемник информации. В качестве приемника информации могут использоваться РОН или ячейка памяти.

Команда POP копирует данные из вершины стека в приемник. В качестве приемника могут использоваться только 16-ти и 32-ух разрядные хранилища! Количество команд PUSH должно соответствовать количеству команд POP.

Примеры:

POP AX Восстановить значение регистра AX из стека.

POP DWord Ptr [EBX] Восстановить содержимое четырех соседних ячеек, первая из которых имеет адрес с номером, хранящимся в EBX, из стека.

Команда POP имеет модификации:

POPA – восстановить значения всех 16-ти разрядных РОН из стека;

POPAD – восстановить значения всех 32-ух разрядных РОН из стека;

**Команда XCHG**

Формат команды: XCHGП,И

где П – приемник информации. В качестве приемника может использоваться РОН;

И – источник информации. В качестве источника информации могут использоваться РОН или ячейка памяти.

Команда XCHG обменивает данные между источником и приемником. Приемник и источник должны быть одинаковой размерности! Нельзя в команде XCHG обменивать данные между двумя ячейками памяти!

**Команда BSWAP**

Формат команды: BSWAPП

где П – приемник информации. В качестве приемника может использоваться только 32-ух разрядный РОН.

Команда BSWAP обменивает в двойном слове между собой значения младшего и старшего байтов, а также средних байтов. Схема обмена представлена на рисунке 11.



Рисунок 11. Схема обмена байтами в команде BSWAP.

**Команда MOVZX**

Формат команды: MOVZXП,И

где П – приемник информации. В качестве приемника может использоваться 16-ти или 32-ух разрядный РОН;

И – источник информации. В качестве источника информации могут использоваться 8-ми или 16-ти разрядные РОН или ячейки памяти.

Команда MOVZX перемещает данные из источника в приемник, увеличивая размерность данных в соответствии с размером приемника, путем добавления нулевых битов слева, как это представлено на рисунке 12.



Рисунок 12. Схема увеличения размерности данных в команде MOVZX.

**Команда MOVSX**

Формат команды: MOVSXП,И

где П – приемник информации. В качестве приемника может использоваться 16-ти или 32-ух разрядный РОН;

И – источник информации. В качестве источника информации могут использоваться 8-ми или 16-ти разрядные РОН или ячейки памяти.

Команда MOVSX перемещает данные из источника в приемник, увеличивая размерность данных в соответствии с размером приемника, путем добавления значения старшего бита источника в расширяемые биты приемника, как это представлено на рисунке 13.



Рисунок 13. Схема увеличения размерности данных в команде MOVSX.

**Команда LEA**

Формат команды: LEAП,И

где П – приемник информации. В качестве приемника может использоваться только 16-ти или 32-ух разрядные РОН. (В операционной системе Windows необходимо использовать 32-ух разрядные РОН);

И – источник информации. В качестве источника информации могут использоваться 16-ти или 32-ух разрядные ячейки памяти.

Команда LEA вычисляет адрес операнда-источника и помещает его в приемник. Вычисление производиться самим процессором во время выполнения программы.

Пример:

1 var

2 Data : Array[1..100] of Byte;

3 begin

4 asm

5 PUSHAD

6 LEA EBX,Data

7 LEA ECX,Data+5

8 MOV Byte Ptr [EBX],$55

9 MOV Word Ptr [ECX],$1234

10 POPAD

11 end;

12 end;

Программа демонстрирует пример совмещения кода на языке высокого уровня и ассемблера. Первые две строчки программы описывают переменную *Data* типа массив на сто байтовых элементов. Третья строка начинает тело программы, а двенадцатая закрывает. Четвертая строка открывает ассемблерную вставку, а одиннадцатая закрывает. Для нормального функционирования программы, которую сформирует компилятор, необходимо запомнить значения всех регистров перед началом выполнения ассемблерной вставки (строка 5), а по ее завершении восстановить (строка 10). Дело в том, что компилятор строго контролирует содержимое регистров, используя их для выполнения программы. Ассемблерная вставка – это код, сразу предназначенный для процессора, и, что в нем будет запрограммировано, компилятор предугадать не может, и, соответственно, не может освободить нужные регистры. Ассемблерная же вставка может изменить значения регистров, а значит, дальнейшее нормальное выполнение программы может быть нарушено. Для этого значения регистров временно сохраняют в стеке, а потом восстанавливают. Шестая строка вычисляет адрес первого байта массива *Data* и сохраняет его в регистре EBX. Дело в том, что операционная система *Windows* может загрузить программу по любому адресу общей памяти. Все будет зависеть от уже загруженных программ, динамических библиотек, состояния системы и т.д. Чтобы использовать массив *Data*, если неизвестно, по какому адресу загружена программа и где выделено место для массива *Data*, нужно в программе вычислить его адрес, для чего и применяется команда LEA. Седьмая строка вычисляет адрес 6-ого байта массива *Data* (у первого байта адрес метки *Data*, у второго *Data+1*, у третьего *Data+2*, и т.д.) и сохраняет его в регистре ECX. Смысл 8-ой и 9-ой строк понятен (см. команду MOV). В результате выполнения программы получим:

Data[1] = 5516 (8510);

Data[6] = 3416 (5210);

Data[7] = 1216 (1810);

**Примерные контрольные вопросы и задачи**

1. Чему будут равны регистры CH, CL, CX после выполнения команды   
   MOV ECX,876543210 ?
2. Чему будут равны регистры AX, BX и EDX после выполнения следующего фрагмента программы:

MOV CL,$127

MOV CH,$128

MOVSX AX,CH

MOVZX BX,CL

MOVSX EDX,BX ?

1. Чему будет равен регистр EDX после выполнения следующего фрагмента программы:

LEA EBX,Data

MOV ECX,$78FA564D

MOV EDX,$12E90BC3

MOV [EBX+3],EDX

MOV [EBX+5],ECX

MOV Word Ptr [EBX+2],$B3

MOV EDX,[EBX+3] ?

1. Что такое дамп памяти?
2. Что такое стек?
3. Назовите все восьмиразрядные РОН?
4. Назовите все шестнадцатиразрядные РОН?
5. Назовите все тридцатидвухразрядные РОН?
6. Чему будут равны регистры BL, BX, BH после выполнения следующего фрагмента программы:  
   MOV EBX,1988776655

BSWAP EBX ?

1. Что означает префикс DWord Ptr и где он используется?
2. Чему будут равны регистры BX и CX после выполнения следующего фрагмента программы:

MOV EAX,$12345678

PUSH EAX

POP BX

POP CX ?

1. Чему равен старший бит числа 200, если его хранить в памяти как слово (Word)?