# 9. Обработка строк. Цепочечные команды 9.1. Команды обработки строк.

# Они позволяют производить действия над блоками байтов или слов памяти. Эти блоки (или строки) могут иметь длину до 64 Кбайт и состоять из числовых или алфавитно-цифровых значений (типа символов в кодах ASCII). Эти команды также называют цепочечными командами . Таким образом, цепочечные команды позволяют проводить действия над блоками памяти, представляющими собой последовательности элементов следующего размера: 8 бит — байт; 16 бит — слово; 32 бита — двойное слово. Всего в системе команд микропроцессора имеется семь операций-примитивов обработки цепочек: **movs cmps lods scas stos ins outs**

# Каждая из них реализуется в микропроцессоре тремя командами, в свою очередь, каждая из этих команд работает с соответствующим размером элемента — байтом, словом или двойным словом.

|  |  |  |
| --- | --- | --- |
| **Команды пересылки:**  movs приемник,источник  movsb  movsw  movsd | **Команды сравнения цепочек:**  cmps приемник, источник  cmpsb  cmpsw  cmpsd | **загрузка элемента из цепочки:**  lods адрес\_источника  lodsb  lodsw  lodsd |
| **сканирование цепочки:** scas адрес\_приемника  scasb  scasw  scasd | сохранение элемента в цепочке:  stos адрес\_приемника  stosb  stoswstosd | **получение элементов цепочки из порта ввода-вывода:** ins адрес\_приемника,номер\_порта  insb insw insd**вывод элементов цепочки в порт ввода-вывода:** outs номер\_порта,адрес\_источника  outsb outsw outsd |

# Особенность всех цепочечных команд в том, что они, кроме обработки текущего элемента цепочки, осуществляют еще и автоматическое продвижение к следующему элементу данной цепочки.

# Предполагается, что строка-приемник находится в дополнительном сегменте ES, а строка-источник – в сегменте данных DS. Процессор адресует строку-приемник через регистр EDI/DI, а строку-источник – через регистр ESI/SI. Перед выполнением команд необходимо обеспечить, чтобы эти регистры содержали смещение первых элементов обеих строк относительно начала соответствующего сегмента.

# Это можно сделать с помощью команды LEA (загрузить исполнительный адрес). Команда LEA пересылает смещение ячейки памяти в любой битовый регистр общего назначения, регистр указателя или индексный регистр. Она имеет формат LEA регистр,память Если обе строки находятся в сегменте данных, то необходимо в регистр ES поместить адрес начала сегмента данных, т.е. значение, находящееся в регистре DS. Это можно сделать так:

|  |  |
| --- | --- |
| push dspop es | mov ax,ds mov es,ax |

# Так как команды манипулирования строками предназначены для действий над *группой* элементов, то они автоматически модифицируют указатели для адресации следующего элемента строки. Бит флага направления DF в регистре флагов определяет, будут значения регистров SI и DI увеличены или уменьшены по завершении выполнения команды.

# Если флаг DF равен 0, то значения регистров ESI/SI и EDI/DI *увеличиваются* после исполнения каждой команды. Если флаг DF равен 1, то они *уменьшаются.* Состоянием флага DF можно управлять с помощью двух команд: CLD – сбросить флаг направления, которая полагает его равным нулю, STD – установить флаг направления, которая присваивает ему значение 1. **9.2. Префиксы повторения**Логически к этим командам нужно отнести и так называемые префиксы повторения. Они предназначены для использования цепочечными командами. Префиксы повторения имеют свои мнемонические обозначения: **rep repe или repz repne или repnz**

# Эти префиксы повторения указываются перед нужной цепочечной командой в поле метки. Цепочечная команда без префикса выполняется один раз. Размещение префикса перед цепочечной командой заставляет ее выполняться в цикле. **Префикс повторения REP** представляет собой не команду, а однобайтовый модификатор, который заставляет микропроцессор выполнить аппаратные повторения команды обработки строк. Это значительно сокращает время на обработку длинных строк по сравнению с программно-организованными циклами. Число повторений извлекается из регистра. После очередного выполнения команды значение в регистре ECX/СХ уменьшается на единицу. **Префикс повторения rep (REPeat).** Этот префикс используется с командами, реализующими операции-примитивы пересылки и сохранения элементов цепочек — соответственно, movs и stos, (ins и outs). Префикс rep заставляет данные команды выполняться, пока содержимое в ecx/cx не станет равным 0. При этом цепочечная команда, перед которой стоит префикс, автоматически уменьшает содержимое ecx/cx на единицу. **Действия rep:** 1. анализ содержимого cx: если cx<>0, то выполнить цепочечную команду, следующую за данным префиксом и перейти к шагу 2; если cx=0, то передать управление команде, следующей за данной цепочечной командой (выйти из цикла по rep); 2.уменьшить значение cx=cx–1 и вернуться к шагу 1; **Префиксы повторения repe или repz (REPeat while Equal or Zero).** Они заставляют цепочечную команду выполняться до тех пор, пока содержимое ecx/cx не равно нулю или флаг ZF равен 1, т.е. пока результат=0 . Как только одно из этих условий нарушается, управление передается следующей команде программы. Благодаря возможности анализа флага ZF, наиболее эффективно эти префиксы можно использовать с командами cmps и scas для поиска отличающихся элементов цепочек.

# **Префиксы повторения repne или repnz (REPeat while Not Equal or Zero).** Префиксы repne/repnz заставляют цепочечную команду циклически выполняться до тех пор, пока содержимое ecx/cx не равно нулю или флаг zf равен нулю. При невыполнении одного из этих условий работа команды прекращается. Данные префиксы также можно использовать с командами cmps и scas, но для поиска совпадающих элементов цепочек. **9.3 Команды пересылки строк** MOVS, MOVSB, MOVSW,MOVSD  Команда MOVS копирует байт или слово из одной части памяти в другую. Она имеет формат MOVS строка\_приемник, строка\_источник Строка­источник – строка в сегменте данных, а строка-приемник – строка в дополнительном сегменте. Команда MOVS копирует один элемент сегмента данных, адрес которого находится в регистре ESI/SI, в ячейку дополнительного сегмента, адрес которой находится в регистре EDI/DI.

# После пересылки элемента команда MOVS изменяет указатели строки-источника ESI и строки-приемника EDI. Если флаг DF равен 0, то микропроцессор увеличивает значения регистров ESI и EDI после пересылки и тем самым адресуется к следующим элементам памяти. Если флаг DF равен 1, то микропроцессор уменьшает значения регистров ESI и EDI после пересылки и тем самым адресуется к предыдущему элементу памяти. Команды MOVSB, MOVSW и MOVSD выполняют аналогичные действия. При этом первая команда оперирует элементами длиной в байт, вторая – элементами длиной в слово (2 байта), а третья – элементами длиной в двойное слово (4 байта). В этих командах не требуются операнды. Они имеют следующий формат:

# При трансляции программы Ассемблер всегда преобразует команду с операндами в одну из команд без операндов. Так как размер элементов – единственная полезная информация, извлекаемая Ассемблером из команды MOVS, то применение команд movsb и movsw movsd предпочтительнее, поскольку в этом случае Ассемблеру нет необходимости выяснять размер операндов.

# Пример1. #include <iostream> using namespace std; void main() {char s1[]="assembler"; char s2[11]; int len=strlen(s1); \_asm { mov ecx,len add ecx,1 cld push ds pop es lea edi,s2 lea esi,s1 rep movsb } cout<<s2; }

# **9.4. Команды сравнения строк.**

# CMPS строка\_приемник, строка\_источник

# CMPSВ CMPSW CMPSD Команда сравнения строк CMPS сопоставляет операнд-источник с операндом-приемником и возвращает результат через флаги. Команда не изменяет значения операндов. *Алгоритм работы:* 1. выполнить вычитание элементов (источник - приемник), адреса элементов предварительно должны быть загружены: адрес источника — в пару регистров ds:esi/si; адрес назначения — в пару регистров es:edi/di; 2. в зависимости от состояния флага df изменить значение регистров esi/si и edi/di: если df=0, то увеличить содержимое этих регистров на длину элемента последовательности; если df=1, то уменьшить содержимое этих регистров на длину элемента последовательности; 3. в зависимости от результата вычитания установить флаги: если очередные элементы цепочек не равны, то cf=1, zf=0; если очередные элементы цепочек или цепочки в целом равны, то cf=0, zf=1; 4. при наличии префикса выполнить определяемые им действия ( repe/repne).

# Команда CMPS сравнивает операнды с помощью их вычитания, а именно *вычитается операнд-приемник из операнда-источника.* Для сравнения нескольких элементов команду CMPS надо использовать с префиксом повторения. В данном случае префикс REP не имеет смысла, т.к. во флагах будет возвращен лишь результат сравнения двух последних элементов. С командой CMPS используют REPE (REPZ) или REPNE (REPNZ).

# С префиксом REPE строки сравниваются до тех пор, пока либо значение регистра СХ не станет равным нулю, либо не будет найдена пара несовпадающих элементов. Пример 2. Поиск пары одинаковых символов

# cld

# push ds pop es mov сx, 10 lea edi,s2 lea esi,s1 repe cmps //; пока = jnz no mov flag,0 jmp \_end no: mov flag,1 \_end: будут сравнивать до 10 пар элементов S1 и S2 с целью найти два несовпадающих элемента

# С префиксом REPNE строки сравниваются до тех пор, пока либо значение регистра СХ не станет равным нулю, либо не будет найдена пара совпадающих элементов. Например, команды cld mov сх, 100 repne смрs dest, source будут сравнивать до 100 пар элементов строк source и dest с целью найти два совпадающих элемента.

# **9.4. Команды сканирования строк**Команды сканирования строк позволяют осуществить поиск заданного значения в строке. При сканировании строки байтов искомое значение должно находиться в регистре AL, а при сканировании строки слов – в регистре АХ, а при сканировании строки двойных слов – в регистре ЕАХ.

# Основная команда группы команд сканирования строк SCAS (scan string – сканировать строку) имеет формат scas строка\_приемник scasb // адрес сканируемой строки esi/si scasw scasd *Алгоритм работы:* 1. выполнить вычитание (элемент цепочки-(eax/ax/al)). Элемент цепочки локализуется парой es:edi/di. Замена сегмента es не допускается; 2. по результату вычитания установить флаги; 3. изменить значение регистра edi/di на величину, равную длине элемента цепочки. Знак этой величины зависит от состояния флага df: df=0 — величина положительная, то есть просмотр от начала цепочки к ее концу; df=1 — величина отрицательная, то есть просмотр от конца цепочки к ее началу

# **Пример 3. Определить число пробелов в строке str** char str[]= “…..”;

# int len= strlen(str); \_asm { mov ax,ds mov es,ax lea edi,str mov ecx,len // ;длину строки - в ecx mov al,' ' mov bx,0 ;счетчик для подсчета пробелов в строке cld

# loop:

# repe scasb jcxz exit ;переход на exit, если ;цепочка просмотрена полностью inc bx jmp loop exit: ...

# **9.5. Команды загрузки строки**

# LODS/LODSB/LODSW/LODSD (Load String Byte/Word/Double word operands) Загрузка строки байтов/слов/двойных слов Данные команды служат для того, чтобы загрузить элемент строки в регистр Команда загрузки формат lods строка\_источник lodsb  lodsw  lodsd *Алгоритм работы:* 1. загрузить элемент из ячейки памяти, адресуемой парой ds:esi/si, в регистр al/ax/eax. 2. Размер элемента определяется неявно (для команды lods) или явно в соответствии с применяемой командой ( lodsb, lodsw, lodsd); 3. изменить значение регистра si на величину, равную длине элемента цепочки. 4.Знак величины зависит от состояния флага df: df=0 — значение положительное, то есть просмотр от начала цепочки к ее концу; df=1 — значение отрицательное, то просмотр от конца цепочки к ее началу. Команды не влияет на флаги

# **9.6. Команды сохранения строки**STOS/STOSB/STOSW/STOSD (Store String Byte/Word/Double word operands) Данные команды служат для того, чтобы загрузить элемент из регистра в память(строку). Команда сохранения строки STOS (Store string – сохранить строку) имеет формат stos строка\_приемник stosb stosw stosd Будучи повторяемой, команда STOS удобна для заполнения строки заданным значением

# *Алгоритм работы:* 1. записать элемент из регистра al/ax/eax в ячейку памяти, адресуемую парой es:di/edi. 2. Размер элемента определяется неявно (для команды stos) или конкретной применяемой командой (для команд stosb, stosw, stosd); 3. изменить значение регистра di на величину, равную длине элемента цепочки. 3. Знак изменения зависит от флага df: df=0 — увеличить, что означает просмотр от начала цепочки к ее концу; df=1 — уменьшить, что означает просмотр от конца цепочки к ее началу. выполнение команды не влияет на флаги

# **Пример 4. Заполнить область памяти пробелами** char str=“'Какая-то строка” int len=strlen(str) \_asm mov ds,ax mov es,ax cld mov al,' ' lea edi,str mov ecx,len rep stosb ;заполняем пробелами строку str

# **Пример 5.Копировать одну строку str1 в другую str2 до первого пробела. Cовместная работа** **stosb и lodsb:** \_asm

# { mov es,ax mov ds,ax cld mov ecx,len // Длина строки lea esi,str1 lea edi,str2 m1: lodsb cmp al,' ' jc exit ;выход, если пробел stosb loop m1 exit: }