Лабораторная работа 1. МАКРОСРЕДСТВА

Макроассемблер имеет специальный набор псевдокоманд, позволяющих программисту пометить одну из секций команд одним именем. Если эта секция встречается в программе много раз, то каждый раз ее можно заменять именем макрокоманды. Использование макрокоманд имеет преимущества перед повторной записью программы: снижается утомительность операции частого переписывания и, соответственно, уменьшается вероятность ошибки; ошибка, обнаруженная в макрокоманде, корректируется только в одном сегменте программы, что уменьшает время отладки; снижаются дублирующие усилия программистов, так как отдельные макрокоманды могут помещаться в системную библиотеку макрокоманд. Макрокоманды позволяют выполнять на языке ассемблера функции языков высокого уровня без необходимости переключаться на другие языки. Кроме того, макрокоманды могут использоваться для повышения читаемости программы. Макрокоманды короче, чем генерируемые ими команды на языке ассемблера, и ближе к естественному языку. Это делает программу более лаконичной и понятной.

Макрокоманда определяется до ее использования псевдокомандой MACRO, имеющей следующий формат:

```
метка код операции операнд
 имя    MACRO   псевдопараметр(ы)
          .
          .
          .
          .
        ENDM
```

Псевдокоманда MACRO содержит имя, по которому позже можно будет обратиться к макрокоманде, и псевдопараметры, заменяемые в процессе расширения макрокоманды. Определение макрокоманды завершается псевдокомандой ENDM. Операторы, ограниченные псевдокомандами MACRO и ENDM, называются телом макрокоманды.

Необязательный псевдопараметр может быть любым определяемым пользователем символическим именем. Если перечисляется больше, чем один параметр, они разделяются запятыми.

В тело макрокоманды могут включаться любые команды МП и допустимые псевдокоманды ассемблера.

Обращение к макрокоманде (макрообращение) производится по ее символическому имени с указанием фактических параметров. Формат макрообращения:

```
  метка          код операции        операнды
необязательная: имя макрокоманды   фактический(ие)
                                     параметр(ы)
```

Фактические параметры должны указываться в том же самом порядке, в каком они перечисляются в макроопределении.

Механизм макрокоманд принципиально отличается от механизма вызова подпрограмм. Вызов подпрограмм осуществляется с дополнительной затратой времени микропроцессора. Когда в машинной программе встречается команда вызова подпрограммы CALL, микропроцессор сохраняет в стеке адрес следующей команды (адрес возврата) и заносит в счетчик команд адрес, заданный в команде CALL. При возврате из подпрограммы по команде RET производится извлечение из стека в счетчик команд адреса возврата. Замена имени макрокоманды на последовательность команд – макрорасширение – реализуется не МП, а ассемблером. Каждый раз, когда в исходном тексте встречается имя макрокоманды, макроассемблер вместо этого имени подставляет последовательность команд макроопределения.

Решение вопроса о том, что использовать – подпрограмму или макрокоманду, в данной конкретной ситуации не всегда однозначно. Например, использование подпрограмм для уменьшения общего размера программы может вызвать значительно более медленное ее выполнение. Длинные программы следует организовывать в виде подпрограмм, в то время как программы, содержащие большое количество параметров, лучше кодировать с использованием макрокоманд.

Псевдокоманды ассемблера для задания макроопределений можно разделить на четыре: группы общего назначения (MACRO, ENDM, LOCAL), повторения (IRP, IRPC, REPT), условные (IF1, IFB, IFNB, EXITM) и управления листингом (.LALL, .SALL, .XALL) (прил. 1, 2).

Рассмотрим пример разработки программы с использованием макроопределений.

Задание: Написать программу, которая изображает "улыбающуюся рожицу" (код символа 2) в столбце 0 случайно выбранной строки, а затем перемещает ее вдоль экрана. "Рожица" должна продвигаться за прием на один столбец, но при этом может перепрыгивать на одну строку вверх или вниз в зависимости от того, что выдает генератор случайных чисел: 0 (вниз на одну строку), 1 (вверх на одну строку) или 2 (та же строка). Операция вывода изображения должна завершиться, если "рожица" пересечет строку 0, строку 24 или столбец 79 [2].

Решение: Текст программы имеет следующий вид (файл macro.asm):

```
; Макроопределения
; Макроопределение задержки на заданное время
delay   macro minutes,seconds,hundr
        local secs,mins,hrs,check,quit
        push_regs <ax,bx,cx,dx>; сохранить регистры
        read_time           ; считать текущее время
        mov     ah,ch     ; скопировать часы в AH,
        mov     al,cl     ;  минуты -в AL,
        mov     bh,dh     ;  секунды -в BH,
        mov     bl,dl     ;  сотые доли -в BL
; получить время конца паузы, добавляя значения
; входных параметров к текущему времени
        ifnb <minutes>
             add al,minutes
        endif
        ifnb <seconds>
             add bh,seconds
        endif
        ifnb <hundr>
             add bl,hundr
        endif
; учесть возможные переносы
        cmp     bl,100
        jb      secs
        sub     bl,100
        inc     bh
secs:   cmp     bh,60
```

```
        jb      mins
        sub     bh,60
        inc     al
mins:   cmp     al,60
        jb      hrs
        sub     al,60
        inc     ah
hrs:    cmp     ah,24
        jne     check
        sub     ah,ah
; ждать наступления полученного времени считыванием
; текущего  времени и сравнением с полученным
check:  read_time
        cmp     cx,ax
        ja      quit
        jb      check
        cmp     dx,bx
        jb      check
quit:   pop_regs <dx,cx,bx,ax> ; восстановить
        endm                   ; значения регистров

; Макроопределение занесения регистров в стек
;в заданном порядке
push_regs macro reg_list
        irp     reg,<reg_list>
        push    reg
        endm
        endm

; Макроопределение восстановления регистров
; из стека в заданном порядке
pop_regs macro reg_list
        irp     reg,<reg_list>
        pop     reg
        endm
        endm

; Макроопределение получения в AL случайного числа
;в промежутке 0 - lim, где lim=4-127
rand    macro lim
        local   strip
        push_regs <cx,dx,ax>; сохранить значения
                            ; регистров
```

```
        mov     ax,0   ; считать показания таймера
        int     1ah
        mov     ax,dx; поместить младшую часть в AX
        mov     cl,lim   ; поместить lim и CL
; удалить из делимого (AX) достаточное число
; старших битов, чтобы гарантировать отсутствие
; переполнения
        mov     dh,3fh; поместить в DH маску
                      ; для операции AND
        cmp     cl,64
        jae     strip
        shr     dh,1  ; если lim<64, удалить 3 бита
        cmp     cl,32
        jae     strip
        shr     dh,1  ; если lim<32, удалить 4 бита
        cmp     cl,16
        jae     strip
        shr     dh,1  ; если lim<16, удалить 5 бит
        cmp     cl,0
        jae     strip
        shr     dh,1  ; если lim<8, удалить 6 бит
        shr     dh,1
strip:  and     ah,dh  ; удалить биты
        div     cl   ; разделить результат в AX
                     ; на lim в  CL
        mov     al,ah; поместить остаток в AL
        pop     cx   ; восстановить значения
        mov     ah,ch; регистров
        pop_regs <dx,cx>
        endm

; Макроопределение чтения текущего времени
;(CH - часы, CL - минуты, DH - секунды, DL – сотые
; доли секунды)
read_time macro
        push    ax
        mov     ah,2ch; выбрать режим чтения
                      ; времени
        int     21h   ; считать время
        pop     ax
        endm
```

```
; Начало программы
code    segment
        assume cs:code
start:  push_regs <ax,bx,cx,dx>  ; сохранить
                                 ;значения регистров
        mov     ah,15 ; загрузить в BH номер
        int     10h   ; активной страницы экрана
        mov     al, 2
        mov     ah,0  ; задать текстовый
        int     10h   ; черно-белый режим 80*25
        mov     cx,1 ; установить счетчик символов
        mov     dl,0  ; установить столбец 0
        rand    24 ; выбрать строку через генератор
        mov     dh,al ; случайных чисел
crsr:   mov     ah,2  ;установить позицию курсора
        int     10h
        mov     al,2  ; задать символ - "рожицу"
        mov     ah,10 ; вывести символ  на экран
        int     10h
        delay   ,   ,12 ; выдержать паузу
        sub     al,al   ; стереть "рожицу"
        mov     ah,10
        int     10h
        inc     dl      ; выбрать следующую позицию
        cmp     dl,80   ; в столбце и выйти, если
        je      ext     ;  он равен 80
        cmp     dh,0  ; проверить не установлены ли
        je      ext   ; граничные значения строки
        cmp     dh,24 ; и если да, то выйти
        je      ext
        rand    60  ; используя генератор случайных
        cmp     al,20 ;  чисел, выбрать направление
        jbe     zero   ;  движения "рожицы": вверх,
        cmp     al,40  ;  вниз, вперед
        jbe     one
        jmp     crsr
zero:   inc     dh     ; двигаться вниз
        jmp     crsr
one:    dec     dh     ; двигаться вверх
        jmp     crsr
ext:    pop_regs <dx,cx,bx,ax> ; восстановить
                               ;регистры
```

```
        mov     ah,4ch              ; выйти в DOS
        int     21h
code    ends
        end     start
```

Задание к лабораторной работе

Составить макроопределение для варианта задания, соответствующего порядковому номеру студента в списке группы. Для проведения отладки макрокоманды составить программу, включающую макрообращение.

1. Макрокоманда моделирования условного оператора Фортрана IF (I.GT.CONSTANT) GO TO LABEL, где I – содержимое ячейки памяти, CONSTANT – непосредственный операнд, LABEL – заданная метка.

2. Макрокоманда удаления значения, находящегося в 8-разрядном регистре DL, из неупорядоченного списка в ОЗУ. При удалении элемента сдвинуть все следующие за ним элементы на одну позицию влево. Начальный адрес списка берется из 16-разрядного регистра BX. Длина списка (в байтах) находится в первой ячейке списка.

3. Макрокоманда деления содержимого ячейки DEND на содержимое ячейки DSR. Операнды считать целыми без знаков (длиной 1 байт) и выполнить деление простым вычитанием делителя из делимого до получения отрицательного результата. Частное и остаток сдублировать в регистре BH и BL соответственно.

4. Макрокоманда добавления значения, находящегося в регистре DL, в конец неупорядоченного списка в ОЗУ (при условии, что такого значения в списке нет). Начальный адрес списка берется 16-разрядный регистр BX. Длина списка (в байтах) находится в первой ячейке списка. Если список пуст, содержимое DL должно становиться первым элементом списка.

5. Макрокоманда формирования дополнительных кодов элементов массива. Начальный адрес массива – ADDR, длина – COUNT.

6. Макрокоманда поиска в упорядоченном списке элемента, равного содержимому ячейки X, и замены его содержимым ячейки Y. Число элементов списка задается первым элементом

списка. Начальный адрес списка находится в 16-разрядном регистре BX.

7. Макрокоманда, аналогичная оператору IF в языке PL/M. Если X больше Y, то Z присваивается значение X, в противном случае Z присваивается значение Y.

8. Макрокоманда формирования обратных кодов элементов массива. Начальный адрес массива – ADDR, длина – COUNT.

9. Макрокоманда аналогичная оператору IF в Фортране для проверки числа в аккумуляторе. Если число отрицательное, то управление передать ячейке NEG, если оно равно 0, управление передать ячейке ZER, если же число положительное, управление передать ячейке POS.

10. Макрокоманда проверки ячейки памяти T. Если содержимое T равно 0, занести содержимое ячейки X в RES, в противном случае занести содержимое ячейки Y в RES.

11. Макрокоманда умножения на число 3. Умножение на 3 = 2 + 1 выполняется за два шага. Вначале содержимое аккумулятора сдвигается на один разряд влево, затем к нему прибавляется множимое.

12. Макрокоманда для определения знака и абсолютной величины операнда, находящегося в ячейке памяти U. Знаковый бит необходимо записать в ячейке SIGN, а абсолютную величину – в ячейку ABS.

13. Макрокоманда сброса бит 0 и 1 регистра RN, если они оба установлены, их установки, если они сброшены, в остальных случаях значения бит не изменяются.

14. Макрокоманда умножения U на V путем сложения V самого с собой U раз. U, V – адреса памяти.

15. Макрокоманда обнуления ячеек с адресами POINT1-POINT2.

16. Макрокоманда формирования обратных кодов элементов массива ОЗУ с адресами POINT1-POINT2.

17. Макрокоманда подсчета количества положительных чисел в ячейках от ADR до ADR+N и запись результата в регистр DL.

18. Макрокоманда формирования дополнительных кодов элементов массива ОЗУ с адресами POINT1-POINT2.

19. Макрокоманда проверки битов 2 и 4 ячейки TOM. Если оба бита установлены, перейти к ERR12, если установлен только

бит 4, перейти к ERR2, если оба бита сброшены, продолжить выполнение программы.

20. Макрокоманда декремента заданного регистра на 1 и перехода к BEGIN, если результат равен LIM.

Порядок выполнения работы

1. Изучить основные сведения по работе.

2. В соответствии с индивидуальным заданием разработать макроопределение и программу на языке ассемблера. В программе производится подготовка параметров, макрообращение и вывод результатов. Подготовить 2-3 варианта исходных данных для отладки программы. Получить результаты ее работы.

3. Выполнить трансляцию, построение кода программы и наблюдать результаты ее работы.

4. Изучить и объяснить листинг программы.

Содержание отчета

1. Цель работы.

2. Текст задания, схема алгоритма, текст программы и результаты ее работы

4. Фрагменты листинга с макрорасширением.

Контрольные вопросы

1. Каким образом макросредства облегчают и ускоряют программирование на языке ассемблере?

2. Сравните использование макрокоманд с вызовом подпрограмм. В чем преимущества и недостатки макрокоманд?

3. Назовите основные псевдокоманды для задания макроопределений.

4. Как обрабатываются ассемблером локальные метки?

СПИСОК РЕКОМЕНДУЕМОЙ ЛИТЕРАТУРЫ

1. Андреева А.А. Основы программирования персонального компьютера на языке ассемблера: лабораторный практикум. Чебоксары. Изд-во Чуваш. ун-та, 2013. 84 с.
2. Брэй Б. Микропроцессоры Intel: 8086/8088…80486, Pentium: пер. с англ. СПб: BHV-Петербург, 2005. 1328 с
3. Зубков С.В. Ассемблер для DOS, Windows и UNIX. М.: ДМК-Пресс, 2013. 638 с.

4. Юров В.И. Assembler: учебник для вузов. СПб.: Питер, 2011. 640 с.
5. Юров В.И. Assembler: практикум: учеб. Пособие для вузов. СПб.: Питер, 2007. 400 с.

ПРИЛОЖЕНИЯ

1. Макросредства ассемблера

Псевдокоманда	Назначение
Псевдокоманды общего назначения	
MACRO	Формат: имя MACRO (список формальных параметров) ... ENDM Присваивает имя последовательности операторов языка ассемблера. Каждое определение MACRO должно завершаться псевдокомандой ENDM
LOCAL	Формат: LOCAL (список формальных параметров) Заставляет ассемблер создать уникальное имя для каждой метки из списка формальных параметров и подставить это имя при каждом вхождении метки в расширение макроопределения
Псевдокоманды повторения	
IRP	Формат: IRP параметр,<список аргументов> ... ENDM Заставляет ассемблер повторять находящиеся между псевдокомандами IRP и ENDM операторы по одному разу для каждого аргумента списка. При каждом повторении производится подстановка очередного аргумента вместо каждого вхождения параметра в блок операторов
IPRC	Формат: IPRC параметр, строка ... ENDM Заставляет ассемблер повторять находящиеся между псевдокомандами IRPC и ENDM операторы по одному разу для каждого символа строки. При каждом повторении производится подстановка очередного символа строки вместо каждого вхождения параметра в блок операторов

Псевдокоманда	Назначение
REPT	Формат: REPT выражение ... ENDM Заставляет ассемблер повторять находящиеся между псевдокомандами REPT и ENDM операторы число раз, определяемое выражением
Условные псевдокоманды	
EXITM	Формат: EXITM Завершает расширение макроопределения в зависимости от результата выполнения условной псевдокоманды
IF1	Формат: IF1 выражение ... ENDIF Выполняет, если ассемблер осуществляет первый проход. Обычно используется для включения с помощью оператора INCLUDE файла с библиотекой макроопределения в исходную программу
IFB	Формат: IFB <аргумент> ... ENDIF Выполняется, если <аргумент> пуст. Угловые скобки обязательны.
IFNB	Формат: IFNB <аргумент> ... ENDIF Выполняется, если <аргумент> не пуст. Угловые скобки обязательны
Псевдокоманды управления листингом	
.LALL	Формат: .LALL Вызывает выдачу полного листинга (включая комментарии) всех расширений макроопределений.
.SALL	Формат: .SALL Исключает текст макроопределений из листинга

Псевдокоманда	Назначение
.XALL	Формат: .XALL Вызывает печать только тех строк макроопределения, которые генерируют объектный код. Этот режим устанавливается по умолчанию

2. Операции в макроопределениях

Операция	Назначение
&	Формат: текст&текст Вызывает конкатенацию (слияние) текста или имени
;;	Формат: ;; комментарий Исключает комментарий из листинга, даже если он выдается по команде .LHLL
!	Формат: !символ Используется в аргументе для указания ассемблеру, что символ надо использовать как литерал, а не как имя
%	Формат: %имя Преобразует имя в число. При расширении макроопределения ассемблер подставляет число вместо имени