МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

кафедра «Информационные системы»

Отчёт

по лабораторной работе №2

по дисциплине «Технические средства информационных систем»

Выполнил:

ст.гр. ИС/б-20-2-о

Филозоп А.Н.

Принял:

Минкин С.И.

Севастополь

2022 г.

2 Лабораторная работа №2
Исследование архитектуры и системы команд 16-разрядного процессора

2.1. Цель работы

Исследовать систему команд, архитектуру и основные блоки процессора Intel 8086 и взаимодействие этих блоков процессора при выполнении команд разных типов. Приобрести практические навыки написания ассемблерных программ и отладки их в эмуляторе микропроцессора — экранным отладчиком типа emu8086.

2.2. Постановка задачи

Запустить эмулятор, выбрать шаблон в формате com и ввести в окне редактора пробную программу, приведенную в приложении А. Исследовать процесс выполнения программы (т.е. проследить изменение содержимого регистров, оперативной памяти и стека). Пояснить в форме комментариев к каждой ассемблерной строке исследуемой программы. Запустить эмулятор, выбрать шаблон в формате com и ввести в окне редактора пробную программу, приведенную в приложении Б. Исследовать и пояснить изменения регистров при выполнении каждой из команд. Рассчитать время выполнения программ.

2.3. Ход выполнения работы

В листинге 1 представлен код предложенной в методических указаниях программы с комментариями.

Листинг 1 — Код программы с комментариями

```
mov ax,0255   ; записать в AX значение 255                          ; + 4
inc ax        ; увеличить значение AX на 1                          ; + 2
add ax, alpha ; прибавить к AX значение alpha                       ; + 16
nop           ; ничего не делать в течение цикла                    ; + 1
```

```
mov bx,ax     ; записать в BX содержимое AX                          ; + 2
dec bx        ; уменьшить значение в BX на 1                         ; + 2
sub bx, beta  ; вычести из значения в BX значение в beta             ; + 16
mov dx, bx    ; запичать в DX значение в BX                          ; + 2
sub dx,10     ; вычесть из DX число 10                               ; + 4
xchg ax,dx    ; поменять местами значения регистров AX и DX          ; + 3
push bx       ; переместить в стек значение BX                       ; + 3
push ax       ; переместить в стек значение AX                       ; + 3
pop cx        ; взять значение из стека и поместить в CX             ; + 8
mov si,cx     ; записать в регистр SI значение CX                    ; + 2
mov di,dx     ; записать в DI значение DX                            ; + 2
mov 0150h,cx ; записать содержимое CX в ячейку памяти по адресу 0150h  ; + 8
shl ax,2      ; сдвинуть биты в AX на 2 позиции влево                ; + 8 + 2 / 2
mov dx, offset hello ; записать в регистр DX значение отступа hello    ; + 4
                     ; от начала сегмента
mov ax,0900h  ; записать в AX число 0900h                            ; + 4
int 21h       ; вызвать прерывание 21h                               ; + 51
mov ax,4c00h  ; записать в AX число 4c00h                            ; + 4
int 21h       ; вызвать прерывание 21h                               ; + 51
ret           ; вернуть управление операционной системе

alpha dw 25
beta dw 32
hello db 'Privet kafedra IS!$'
```

Время выполнения программы было высчитано следующим образом. Количество тактов при выполнении каждой команды было сложено, получено значение 201 такт. Так как частота тактового генератора равна 5МГц, то время выполнения программы равно $201 \cdot \frac{1}{5 \cdot 10^9} = 40{,}2 \cdot 10^{-9} c$.

В листинге 2 приведён код второй предложенной программы.

Листинг 2 — Код программы

```
org 100h ;Программа начинается с адреса 100h
jmp start ;Безусловный переход на метку start
;-- Данные ------------------------------------------------------------
v dw 12345
pak db 13,10,'Press any key...$'
;----------------------------------------------------------------------
start:
mov bx,[v] ;BX = v
mov ah,2 ;Функция DOS 02h - вывод символа
mov cx,16 ;Инициализация счѐтчика цикла
lp:
shl bx,1 ;Сдвиг BX на 1 бит влево
mov dl,'0' ;dl = '0'
jnc print ;Переход, если выдвинутый бит равен 0
inc dl ;dl = dl + 1 = '1'
print:
int 21h ;Обращение к функции DOS 02h
loop lp ;Команда цикла
mov ah,9 ;\
mov dx,offset pak ; > Вывод строки 'Press any key...'
int 21h ;/
mov ah,8 ;\
int 21h ;/ Ввод символа без эха
mov ax,4C00h ;\
int 21h ;/ Завершение программы
```

Вывод

При выполнении данной работы было получено понимание работы 16 разрядного процессора Intel8086. Также было получены навыки комментирования и вычисления времени выполнения программы, написанной на языке ассемблера.

Приложение А

Ответы на контрольные вопросы

1. Процессор Intel 8086 содержит два основных блока:
    1. BIU (Bus Interface Unit), предназначенный для считывания команд и их операндов с системной шины; запись данных в память; вычисления физического адреса; сохранения считанных байтов в шестибайтовом конвейере команд;
    2. EU (Execution Unit), предназначенный выполнения команд, в т.ч. арифметических операций.
2. Повышение быстродействия работы процессора 8086 обосновано следующим:
    1. блоки BIU и EU работают параллельно. Как следствие блок исполнения команд в большинстве случаев не ожидает получения команды из оперативной памяти, что способствует увеличению количества выполненных команд за промежуток времени.
    2. повышения максимальной частоты тактового генератора до 5МГц.
3. Типы машинных циклов:
    1. извлечение кода команды;
    2. чтение данных из памяти;
    3. запись данных в память;
    4. извлечение из стека;
    5. запись данных в стек;
    6. ввод данных из внешнего устройства;
    7. запись данных на внешнее устройство;
    8. обслуживание прерывания

4. Описание основных выходов Intel8086 приведено в таблице ниже:

№ контакта	Обозначение	Предназначение
1, 20	GND	Земля
40	V_{cc}	Питание 5В.
2-16	AD0-AD14	Ввод/вывод данных в течение машинного цикла.
17	*NMI*	Входящий запрос немаскируемого прерывания.
18	*INTR*	Входящий запрос маскируемого прерывания.
19	*CLK*	Вход тактового генератора.
21	*RESET*	Вход на сброс состояния регистров.
22	*READY*	Вход из системной шины. Подтверждение шиной завершения ввода/вывода данных.
33	$MN/\bar{M}$	Вход, переключающий работу процессора в минимальный (при подаче +5В) или максимальный (при заземлении).

5. Флаги процессора.

Название флага	Описание
Флаг переноса CF	Устанавливается в 1, если при выполнении сложения/вычитания возник перенос/заём из старшего бита.
Флаг паритета PF	Устанавливается в 1, если младшие 8 байт результата выполнения операции содержит чётное количество единиц.
Флаг вспомогательного переноса AF	Устанавливается в 1, если в результате сложения/вычитания произошёл перенос/заём из третьего бита.
Флаг нуля ZF	Устанавливается в 1, если результат операции равен нулю.
Флаг знака SF	Равен старшему биту результата операции.
Флаг прослеживания TF	Если равен 1, после выполнения каждой команды возникает внутреннее прерывание.
Флаг разрешения прерывания IF	Если равен 1, процессор обрабатывает маскируемые прерывания, иначе игнорирует их.
Флаг направления DF	Применяется в командах манипуляций цепочками.
Флаг переполнения OF	Устанавливается в 1, если результат выполнения операции сложения/вычитания находится за пределами допустимого диапазона.

6. Физический адрес сегмента памяти представляет 20-битовое число, которое однозначно определяет положение каждого байта. Логический адрес ячейки памяти состоит из двух 16-битовых чисел: начального адреса сегмента, который называют сегментом, и смещения, определяемого как расстояние от начала сегмента до этой ячейки.

Для вычисления физического адреса база сегмента сдвигается на 4 бита влево и суммируется со смещением. Возможный перенос из старшего бита игнорируется.

7. В ассемблерных программах применяются команды побитового сдвига влево/вправо (*SHL* и *SHR* соответственно) и команды побитового арифметического сдвига влево/вправо (*SAL* и *SAR* соответственно).
8. Инициализация сегментных регистров происходит путём установки конкретного значения в данных регистр.
9. Стек — область памяти, работающая по принципу «первый вошёл — последний вышел» предназначенная для хранения временных данных. В программах формата EXE программист должен задавать сегмент стека, в программах COM стек генерируется автоматически.
10. Указатель стека содержит смещение ячейки памяти, в которой находятся данные, относительно начала сегмента. При выполнении команды PUSH значение регистра SP уменьшается на 1 машинное слово, а в ячейку по полученному смещению записываются данные.
11. ….
12. При включении компьютера генерируется сигнал на линии сброса *RESET* заставляет все компоненты перейти в выключенное состояние. В процессоре регистры PSW, IP, DS, SS, ES и очередь команд сбрасываются, а в CS загружается код FFFF. После чего процессор начинает выполнение команд с адреса $FFFF0+0000=FFFF0$. Обычно эта ячейка находится в постоянном запоминающем устройстве и содержит команду перехода *JMP* к программе инициализации системы (загрузчику) и загрузки прикладной программы или операционной системы.
13. ….
14. Внутренние прерывания DOS используются для вызова системных функций.
15. ….