Выполнил студент 213 группы: Тихонов Евгений

Ответы на вопросы:

1) Ветвления в алгоритмах. Реализация ветвлений на языке ассемблера.

Ответ: Ветвление в языке ассемблера реализуется с помощью команд условного перехода, которые способны перенести выполнение программы на определенную метку в зависимости от состояния регистра флагов.

2) Команды условных и безусловного переходов. Каким образом вычисляются адреса переходов?

Ответ: JMP OPR - команда безусловного перехода. Команд условного существует большое количество, каждая команда срабатывает по поднятию определенных флагов(комбинаций флагов), например: JE OPR(сработает при ZF = 1), JNE(сработает при ZF = 0), JB/ JNAE OPR(сработает при CF = 1). Режимы адресации условных переходов - относительно программного счетчика (IP). Операнд OPR должен быть в пределах от-128 до127 байт от команды, находящейся за командой перехода.

3) Циклы в алгоритмах. Организация циклов на языке ассемблера. Особенности цикла LOOP.

Ответ: Циклы можно реализовывать с помощью команд условного перехода и метки которая находится выше, в начале цикла. Если по прошествию итерации цикла условие оператора условного перехода выполнится то исполнение начнется с начала цикла, если нет то цикл завершится. Так же можно реализовать цикл с помощью LOOP OPR, которая переходит на метку и каждую итерацию отнимает от регистра счетчика 1. Если регистр счетчика = 0, то команда не сделает переход по метке. Поэтому перед меткой, обозначающей начало цикла стоит задать CX числом обозначающим количество итераций цикла.

4) В каком регистре находится во время выполнения программы смещение

кода? Каким образом вычисляется адрес команды?

Ответ: Регистр EIP содержит смещение следующей к выполнению команды. Этот регистр недоступен программисту, но изменение его значения производятся командами управления, к которым относятся команды условных и безусловных переходов, вызова процедур и возврата из процедур. Адрес следующей выполняемой команды вычисляется относительно начала сегмента (указатель на начало сегмента лежит в CS) с помощью смещения (Смещение может быть 8 или 16 битное), за исключением косвенного перехода, когда адрес следующей на исполнение команды непосредственно попадает в IP, и лишь затем CS присваивается новое значение.

5) Какую принципиальную роль играет оператор безусловного перехода JMP при организации ветвлений?

Ответ: С помощью безусловного оператора можно обходить ветви кода исполнение которых не нужно, так как мы уже исполнили нужную ветку кода.

6) Что означает корректное завершение программы?

Ответ: программа выполнилась без ошибок и может передать управление ОС.

7) Реальный и защищённый режимы работы процессора (на примере Intel 8086). Вычисление физических адресов ячеек памяти.

Ответ: Реальный режим - обращение к оперативной памяти происходит по реальным (действительным) адресам. Набор доступных операций не ограничен, защита памяти не используется.

Защищённый режим-обращение к памяти происходит по виртуальным адресам с использованием механизмов защиты памяти. Набор доступных операций определяется уровнем привилегий. После инициализации

процессор находится в реальном режиме. Процессор может быть переведен в защищенный режим установкой бита 0 (Protect Enable) в регистре CR0:

MOV EAX,00000001h

MOV CR0,EAX

Вернуться в режим реального адреса процессор может по сигналу RESET или сбросив бит PE:

MOV EAX,00000000h

MOV CR0,EAX

Программа:

Программа написана на основе ассемблера TASM.
Тестирование проводились в среде DOSBox, программой TURBO DEBUGGER.

Код программы:

```
stack_seg segment stack "stack"
    db 20 dup(?)
stack_seg ends


data_seg segment
    a db 15h;21
    b db 38h;56
    nod db 0;will be 7, sum of bits = 3
data_seg ends


code_seg segment "code"
    assume ss:stack_seg, ds:data_seg, cs:code_seg
    Start:
```

```
;set data segment
mov ax,data_seg
mov ds,ax
;find NOD (GCD)
mov dh, a
mov dl, b
cmp dh, dl
;CMP a, b
JNAE if_b_max
if_a_max:
    MOV AL, a
    MOV BL, b
    JMP end_if_b_max
if_b_max:
    MOV AL, b
    MOV BL, a
end_if_b_max:

while_AH_not_0:
    MOV AH, 0
    DIV BL
    CMP AH, 0
    JE while_end

    MOV AL, BL
    MOV BL, AH
    JMP while_AH_not_0
while_end:
MOV nod, BL
```

```
        ;get sum bits of nod(write in DX)
        MOV DX, 0
        MOV CX, 8; 8 loop iters, because byte has 8 bit
        loop_start:
            ror nod, 1; right cycle shift
            LAHF; move last 8 FLAGS bits to AH
            and AH, 00000001b; get first bit(CF), there is our shifted
bit from nod
            add DL, AH
            mov AH, 0
            loop loop_start

        ; correct exit
        mov ah,4ch
        int 21h
code_seg ends
end Start
```