Выполнил студент 213 группы: Тихонов Евгений

Ответы на вопросы:

1)Виды межсегментных переходов. Способы вычисления адресов переходов.

Ответ: Межсегментные переходы реализуются только командами безусловных переходов. Существует прямой и косвенный межсегментные переходы. Прямой переход: JMP FAR PTR OPR, где OPR метка. Косвенный переход: JMP OPR, где OPR переменная хранящая адрес. Вычисляться адреса переходов могут относительным режимом(относительно начала сегмента) и прямым режимом(адрес напрямую попадает в IP).

2) Флаги процессора и их использование в условиях.

Ответ: регистр флагов FLAGS имеет множество флагов таких как CF(Carry Flag) - 1, когда случается беззнаковое переполнение,

ZF(Zero Flag) - устанавливается в 1, если результат равен нулю,

SF(Sign Flag) - установлен в 1, если результат отрицательное число,

OF(Overflow Flag) - устанавливается в 1, если случается переполнение при

арифметических операциях со знаком и другие флаги. Операция сравнения CMP влияет на флаги вследствие чего мы можем использовать операцию условного перехода после операции CMP. Исходя из разных требований к флагам используются разные операции условного перехода.

3) Команды линейного логического и арифметического сдвигов. В чем заключается разница их выполнения?

Ответ: Логический сдвиг влево/вправо: SHL OPR,CNT/SHR OPR,CNT. Арифметический сдвиг влево/вправо: SAL OPR,CNT/SAR OPR,CNT. Где OPR сдвигаемый операнд, CNT количество сдвигаемых бит. SAL не сохраняет знака, но устанавливает флаг OF в случае смены знака

очередным выдвигаемым битом. В остальном полностью аналогична команде SHL. SAR сохраняет знак, восстанавливая его после сдвига каждого очередного бита. В остальном – аналогична команде SHR

4) Особенности выполнения команд циклического сдвига. Сферы применения этих команд.

Ответ: Сдвинуть циклически влево: ROL OPR,CNT. Сдвинуть циклически вправо: ROR OPR,CNT. Где OPR сдвигаемый операнд, CNT количество сдвигаемых бит. Выдвинутый бит копируется в CF, затем переносится в начало сдвигаемого операнда. Данную команду можно использовать в подсчете суммы бит в операнде.

5) Что указывает директива ASSUME в программе?

Ответ: ASSUME директива указывает ассемблеру какой сегмент вы «привязываете» к сегментному регистру, чтобы потом получить доступ к нему через этот регистр.

6) Как оформляется начало выполнения программы?

Ответ: Начало программы начинается с директивы ASSUME, где мы привязываем сегменты к сегментным регистрам. Затем обычно идет указание метки(begin, start, main), называемой точкой входа в программу, затем идет инициализация сегмента данных, через регистр AX мы задаем значение регистру данных DS.

Программа:

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

Код программы:

```
;stack segment

s_s segment stack "stack"
```

```
 db 20 dup(?)
s_s ends

;first data segment
d_s1 segment
 aa db 00000111b
d_s1 ends

c_s2 segment "code"
assume ss:s_s,cs:c_s2
jmp m2
c_s2_strt:
assume ss:s_s,ds:d_s1,cs:c_s2
;change data segment
 mov ax,d_s1
 mov ds,ax

 ;add 3rd bit in CF
 ror aa, 3

 LAHF;move last 8 FLAGS bit to ah
 mov bl, ah
 and bl, 00000001b;get first bit(CF)

 jmp FAR PTR ext
m2:
c_s2 ends
```

```
c_s1 segment "code"
assume ss:s_s,cs:c_s1
jmp m1
c_s1_strt:
assume ss:s_s,ds:d_s3,cs:c_s1
 ;change data segment
 mov ax,d_s3
 mov ds,ax

 ;add 5rd bit in CF
 ror bb, 5

 LAHF;move last 8 FLAGS bit to ah
 mov bh, ah
 and bh, 00000001b;get first bit(CF)

 jmp FAR PTR c_s2_strt

m1:
c_s1 ends

;second data segment
d_s2 segment
 code_segment1 dd c_s1_strt
d_s2 ends

;third data segment
d_s3 segment
 bb db 01001010b
```

```
d_s3 ends



c_s3 segment "code"; third code segment
 assume ss:s_s,ds:d_s1, cs:c_s3
begin:
 ;set data segment
 mov ax,d_s1
 mov ds,ax


 ;*2
 sal aa, 1


 ;change data segment
 assume ds:d_s3
 mov ax,d_s3
 mov ds,ax


 ;/4
 sar bb, 2


 ;change data segment
 assume ds:d_s2
 mov ax,d_s2
 mov ds,ax


 jmp [code_segment1]


ext:; correct exit
```

```
 mov ah,4ch
 int 21h


c_s3 ends
end begin
```