Выполнил студент 213 группы: Тихонов Евгений

Ответы на вопросы:

1) Понятие сегмента, характеристики сегмента, организация сегмента.

Ответ: Сегмент - это область памяти характеризуемая начальным адресом и длинной. Длина сегмента - это количество ячеек памяти в нем. Начальный адресс сегмента - адрес в памяти с которого начинается сегмент. Доступ к ячейкам памяти сегмента осуществляется с помощью смещения относительно начального адреса.

2) На какие сегменты разбита память компьютера? В какие регистры записываются начальные адреса сегментов?

Ответ: Обычно в программе используются 3 сегмента: сегмент данных, сегмент стека и сегмент кода. Указатели на данные сегменты хранятся в регистрах процессора: ds(сегмент данных), ss(сегмент стека), cs(сегмент кода).

3) Какие регистры микропроцессора используются при выполнении арифметических операций?

Ответ: Для выполнения арифметических операций можно использовать почти любой регистр, но в основном используют регистры такие как AX, CX, DX, BX. Стоит заметить, что выполнение некоторых арифметических операций(например деления и умножения) возможно только с регистром AX.

4) На какие флаги воздействуют арифметические команды?

Ответ: 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,
PF(Parity Flag) - этот флаг устанавливается в 1, если в младших 8-битовых данных четное число,
AF(Auxiliary Flag) - установлен в 1, если случилось переполнение без знака младших 4-х битов.

5) Какие режимы адресации могут применяться для доступа к данным при выполнении арифметических и поразрядных логических операций?

Ответ: Непосредственный, прямой, регистровый, регистровый косвенный, регистровый относительный, базовый индексный, относительный базовый индексный.

6) Особенности выполнения операции умножения. Особенности выполнения операции деления. Распределение регистров.

Ответ: Операции умножения вызываются с одним операндом, предполагается, что второй операнд находится в рег. AX. Результат умножения сохраняется в тот же регистр но старше. (для бита это AX, для слова EAX и тд.). При делении остаток и частное сохраняются в младшие регистры(для слова остаток в AH, частное в AL)

7) Основные логические операции и принципы их выполнения.

Ответ: AND - побитовое логическое И. OR - побитовое логическое ИЛИ. XOR - побитовое ИСКЛЮЧАЮЩЕЕ ИЛИ. Все эти операции проводятся над 2-мя операндами, результат сохраняется в первый операнд.

NOT - логическое отрицание над единственным операндом. TEST - побитовое логическое И, но результат не сохраняется в первый операнд, при этом поднимаются соответствующие флаги.

8) Правила формирования масок для установки и сброса битов.

Ответ: Для установки битов ым используем операцию OR и в качестве второго операнда используем 2-ичное значение с n нулевыми битами(где n - битная размерность первого операнда: для байта 8бит, для слова 16бит и т.д.), затем в разряды где мы хотим установить биты(установить для первого операнда) ставим единицы(ставим во втором операнде). Со сбросом обратная ситуация. Используем операцию AND и n-размерное двоичное значение в качестве 2-ого операнда, состоящее из единиц и нулей в тех разрядах где мы хотим обнулить биты.

9) Каким образом выполняются логические команды над

Словами?

Ответ: Работа OR и AND для слов не отличается, а работа XOR требует разбиения слова на 2 байта, старшего и младшего, а затем выполнение операции над ними иначе флаг четности PF установится некорректно.

Программа:

Программа написана на основе ассемблера NASM.

Написание и тестирование проводились на системе Linux. Ассемблирование проводилось утилитой nasm 2.15.05 с помощью команды в терминале:

nasm -f elf32 1.1.asm

Линковка программы производилась gcc 9.3.0 с помощью команды в терминале:

gcc -m32 -o myprog 1.1.o

Запуск программы из под терминала:

./myprog

Код программы:

```
; Declare some external functions
extern  printf      ; the C function, to be called

SECTION .data       ; Data section, initialized variables

a:  db  10      ; int a=10;
b:  db  27      ; int b=27;
c:  db  00000101b ; binary 101(2) == 3(10);

sum: db 0
dif: dw 0
prd: dw 1
ost: db 0; ostatok
qtn: db 0; chastnoe

outstr:    db "a=%d, b=%d, sum=%d, dif=%d, prd=%d, qnt=%d, ost=%d", 10, 0
; format string, "\n",'0'

SECTION .text                   ; Code section.
```

```
global main     ; the standard gcc entry point
main:               ; the program label for the entry point
   push    ebp     ; set up stack frame
   mov     ebp,esp

   ;sum
   mov eax, 0
   mov al, [a]  ; put a from mem into register
   add al, [b]  ; a+b
   mov [sum], eax ; put to sum(in mem) from register

   ;diff
   mov eax, 0
   mov ax, [a]
   sub ax, [b]
   mov [dif], ax

   ;diff with neg
   mov eax, 0
   mov ebx, 0
   mov ax, [a]
   mov bx, [b]
   neg bx
   sub ax, bx
   mov [dif], ax

   ;multiply with sign
   mov eax, 0
   mov al, [b]
   neg al
   imul byte[a]
   mov [prd], ax

   ;multiply without sign
   mov eax, 0
   mov al, [b]
   neg al
   mul byte[a]
   mov [prd], ax

   ;div
   mov eax, 0
   mov ax, [b]
   div byte[a]
```

```
mov [ost], ah
mov [qtn], al

;other task
mov al, [qtn]
or al, 11000000b; add 7,8 bits
and al, 10111100b; del 1,2,7 bits

;mod2
mov bl, al
xor al, bl

;console out

;push to stack
;revers push arguments because stack

mov eax, 0

mov al, [ost]
push eax

mov al, [qtn]
push eax

mov eax, 0

mov ax, [prd]
push eax

mov ax, [dif]; push value of variable dif
push    eax;

mov eax, 0

mov al, [sum]; push value of variable sum
push    eax;

mov al, [b]; push value of variable b
push    eax;

mov al, [a]; push value of variable a
push    eax;

push    dword outstr    ; address of format string
```

```
call    printf      ; Call C function
add     esp, 32 ; pop stack 8 (push) times 4 bytes 8*4=32

mov     esp, ebp    ; takedown stack
pop     ebp     ; same as "exit"

mov eax,0       ;  return normal value , no error
ret         ; return
```