Семинар 09.03.2021

Разбор контрольной

ФИО _____

1. Указать какое знаковое десятичное число представляет байт 88h в прямом коде и какое - в дополнительном коде. *Ответ:* 136 : -120

2. Указать значения регистров AH и AL (как десятичных знак. и б/зн. чисел) и флагов CF и ZF после выполнения следующих команд:

а)
```
MOV AH, 0
MOV AL, 160
ADD AL, 160
ADC AH, 3
```
Ответ а) AL = 64 : 64
AH = 4 : 4
CF=0 ZF=0

б)
```
MOV AH, 20
MOV AL, 10
SUB AL, 16
SBB AH, 0
```
Ответ б) AL = 250 : -6
AH: 255 → 79 : -1
CF=0 ZF=0

3 Указать значения регистра CL (как знакового, так и беззнакового десятичных чисел) и флагов CF, OF, SF и ZF после выполнения следующей пары команд:

а)
```
MOV CL,246
ADD CL,216
```
Ответ а) CL = 206 : -50
CF=1 OF=0
SF=1 ZF=0

б)
```
MOV CL,40
SUB CL,100
```
Ответ б) CL = 196 : -60
CF=1 OF=0
SF=1 ZF=0

4. Могут ли после выполнения команды ADD флаги получить следующие значения:
CF=0, OF=1, SF=0 *Ответ(да\нет?):* нет *Пояснение(на обороте):*

ФИО _____

1. Указать какое знаковое десятичное число представляет байт 99h в прямом коде и какое - в дополнительном коде. *Ответ:* 153 : -103

2. Указать значения регистров AH и AL (как десятичных знак. и б/зн. чисел) и флагов CF и ZF после выполнения следующих команд:

а)
```
MOV AH, 0
MOV AL, 150
ADD AL, 60
ADC AH, 3
```
Ответ а) AL = 210 : -46
AH = 3 : 3
CF=0 ZF=0

б)
```
MOV AH, 255
MOV AL, 255
ADD AL, 1
ADC AH, 0
```
Ответ б) AL = 0 : 0
AH : 0 : 0
CF=1 ZF=1

3 Указать значения регистра CL (как знакового, так и беззнакового десятичных чисел) и флагов CF, OF, SF и ZF после выполнения следующей пары команд:

а)
```
MOV CL,128
ADD CL,128
```
Ответ а) CL = 0 : 0
CF=1 OF=1
SF=0 ZF=1

б)
```
MOV CL, -10
ADD CL, -40
```
Ответ б) CL = -50 : 206
CF=1 OF=0
ZF=0 SF=1

4. Могут ли после выполнения команды ADD флаги иметь следующие значения:
CF=OF=SF=1 Ответ(да\нет?): нет Пояснение(на обороте):

Задача

Пример 5-1 Восстановление типа и операции сравнения

Ниже приведен код на языке Си, где data_t – некоторый целочисленный тип данных, а COMP – некоторая операция сравнения. Пусть переменная a расположена в регистре EDX, а переменная b в регистре ECX. По заданному ассемблерному коду, реализующему данную операцию сравнения, требуется восстановить data_t и COMP (возможно несколько вариантов решения).

```
int comp(data_t a, data_t b) {
   return a COMP b;
}
```

```
CMP ECX, EDX
SETL AL
```

Цикл do-while

```
static int n;
int i = 0;
do {
    i++;
} while (i < n)
```

```
xor ecx, ecx ; i = 0
L:
    inc ecx ; i++
    cmp ecx, dword[n] ; i < n
    jl L
```

Цикл while

```
static int n;
int i = 0;
while (i < n) {
    i++;
}
```

```
xor ecx, ecx ; i = 0
L:
    cmp ecx, dword[n] ; i < n
    jnl EXIT
    inc ecx ; i++
    jmp L
EXIT:
```

Цикл for

```
static int n;
static int sum;
for (int i = 0; i < n; ++i) {
    sum += i;
}
```

```
xor ecx, ecx ; i = 0
L:
    cmp ecx, dword[n] ; i < n
    jnl EXIT
    add dword[sum], ecx ; sum += i
    inc ecx ; i++
    jmp L
EXIT:
```

Вложенные циклы

```
static int n;
static int sum;
for (int i = 0; i < n; ++i) {
    for (int j = 0; j < n; ++j) {
        sum += i + j;
    }
}
```

```
xor ecx, ecx ; i = 0
L_OUTER:
    cmp ecx, dword[n] ; i < n
    jnl EXIT_OUTER

    xor edx, edx ; j = 0
    L_INNER:
        cmp edx, dword[n] ; j < n
        jnl EXIT_INNER
        add dword[sum], ecx ; sum += i
        add dword[sum], edx ; sum += j
        inc edx ; ++j
        jmp L_INNER
    EXIT_INNER:

    inc ecx ; ++i
    jmp L_OUTER
EXIT_OUTER:
```

loop

Таблица 1. Описание команд аппаратной поддержки цикла.

Команда	Описание
JCXZ/JECXZ	Переход выполняется, если значение регистра CX/ECX равно нулю.
LOOP	Переход выполняется, если значение регистра ECX не равно нулю.
LOOPZ/LOOPE	Переход выполняется, если значение регистра ECX не равно нулю и флаг ZF установлен.
LOOPNZ/LOOPNE	Переход выполняется, если значение регистра ECX не равно нулю и флаг ZF сброшен.

Выполнение команды LOOP/LOOP*cc* уменьшает значение ECX на единицу, после чего делается условный переход, если выполняется соответствующее условие. Если ECX изначально быть равен 0, то после выполнения команды из семейства LOOP (и передачи управления на метку цикла), ECX станет 0xFFFFFFFF, что (скорее всего) приведет к ошибочной работе программы.

Еще одной особенностью является то, что условный переход в командах LOOP/LOOP*cc* может быть только в пределах [-128, 127] байтов от текущей позиции в коде. Закодировать цикл с достаточно объемным телом не получится, т. к. ассемблер не сможет построить код для инструкции, выполняющей переход на слишком удаленную метку.

Цикл while через loop

```
static int n;
static int sum;
int i = n;
while (i > 0) {
    sum += i;
    i--;
}
```

```
mov ecx, dword[n]
jecxz EXIT
L:
    add dword[sum], ecx
    loop L
EXIT:
```

Задача

По введённому числу N вычислить факториал N и вывести его на печать.

Указатели - адресация

Статические переменные располагаются в одной из секций статических данных; если у переменных не происходит инициализация при объявлении, то расположить их допустимо в секции .bss. В языке ассемблера адрес статической переменной – ее символическое имя. Это имя необходимо присвоить переменной xp.

```
static int *xp;
static int x;
xp = &x;
```

```
section .bss
   xp resd 1
   x  resd 1

section .text
   mov    dword [xp], x
```

Указатели - разыменование

```
static int *xp;
static int x, y;

x = *xp;

*xp = y;
```

```
section .bss
   xp resd 1
   x  resd 1
   y  resd 1

section .text
   mov edx, dword [xp]   ; помещаем в EDX значение переменной xp
   mov eax, dword [edx]  ; помещаем в EAX значение ячейки
                         ; памяти, на которую ссылается xp
   mov dword [x], eax    ; присваиваем это значение x

   mov eax, dword [y]    ; помещаем в EAX значение переменной y
   mov dword [edx], eax  ; в регистре EDX уже находится значение
                         ; переменной xp. Это значение
                         ; используется как адрес, по которому
                         ; будет записано содержимое EAX
```

Указатели - двойное разыменование

Дана статическая переменная p:

```
static int **p;
```

Требуется на языке ассемблера написать фрагмент программы, который вычисляет выражение **p + 1 и печатает его значение на стандартный вывод, используя макрокоманду PRINT_DEC.

Решение

Заданное выражение предполагает двойное разыменование указателя и увеличение полученного значения на единицу.

```
%include 'io.inc'

section .text
global CMAIN
CMAIN:
   MOV  EAX, DWORD [p]    ; Записываем в регистр EAX значение
                          ; переменной p
   MOV  EAX, DWORD [EAX]  ; Используем это значения для доступа
                          ; к памяти, теперь в регистре EAX
                          ; не p, а *p
   MOV  EAX, DWORD [EAX]  ; Повторяем – теперь в EAX находится **p
   INC  EAX               ; Увеличиваем это значение на единицу
   PRINT_DEC EAX          ; Печатаем
   XOR EAX, EAX           ;
   RET                    ;
```

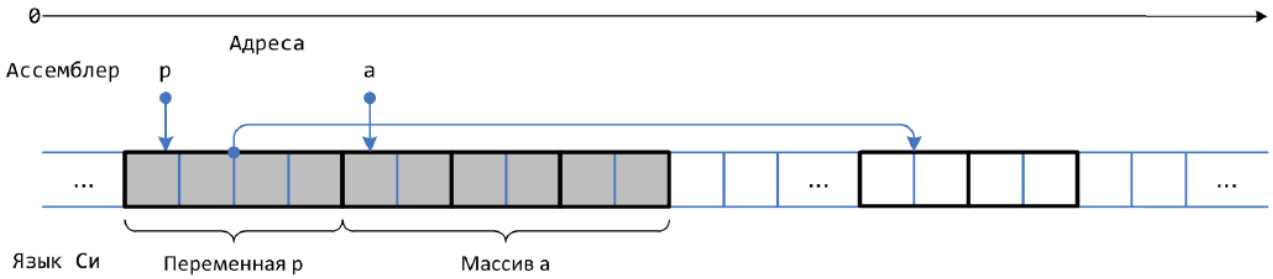
Указатели и массивы - адресная арифметика

Для приведенного фрагмента Си-кода требуется написать соответствующий фрагмент ассемблерной программы.

```
    static short *p;
    static short a[3];
…
    p[1] = *(a + 2);
```

Решение

В приведенном фрагменте массив используется как указатель, а указатель – как массив, с индексным выражением. Тем не менее, ассемблерный код будет отражать особенности фактического выделения памяти. На рисунке ниже показано распределение данных в памяти: черные линии показывают группировку байт в базовые типы, серым цветом показаны выделенные байты. Снизу памяти данные подписаны в терминах языка Си, сверху – адреса, используемые в ассемблерном коде.



Указатели и массивы - адресная арифметика

Память выделена для указателя p, но не для тех ячеек, на которые он указывает. Поэтому, что вычислить адрес первого элемента последовательности int-ов, на которые указывает p, необходимо загрузить адрес (значение переменной p) из памяти в регистр. В случае с массивом память была выделена для всех элементов, а имя массива интерпретируется как адрес начала выделенной памяти. Поэтому при извлечении элемента с индексом два имя массива следует сразу же использовать в адресном коде в качестве базы.

```
mov   dx, word [a + 4]     ; *(a + 2) – то же, что и a[2]
                           ; a – адрес, начиная с которого
                           ; в памяти размещены элементы массива
mov   eax, dword [p]       ; Значение переменной p – адрес,
mov   word [eax + 2], dx   ; который указывает на начало
                           ; массива
```

Задача

Для приведенного фрагмента Си-кода требуется написать соответствующий фрагмент ассемблерной программы.

```
static int *p[10];
static int x;
x = *p[8] + 1;
```

Задача

Заданы два массива целых чисел: массив `short A[100]` и массив `int B[100]`. Требуется выполнить копирование всех элементов массива `A` в массив `B` со знаковым расширением.

Задача

Написать программу, которая вводит число типа `unsigned long long` и печатает в порядке убывания все десятичные цифры, входящие в это число.

Многомерные массивы

```
static int A[M][N];
static int B[N][M];

for (int i = 0; i < M; i++) {
   for (int j = 0; j < N; j++) {
      B[j][i] = A[i][j];
   }
}
```

```
section .bss
   A resd M*N
   B resd N*M

section .text

   mov  esi, 0 ; индекс i

 .l1:                          ; Начинается тело внешнего цикла
   mov  edi, 0 ; индекс j

 .l2:                          ; Начинается тело внутреннего цикла
   imul ecx, esi, N            ; Смещаемся на esi строк по N элементов каждая
   add  ecx, edi               ; … и еще на edi элементов (массив A)
   mov  eax, dword [A + 4*ecx] ; Считываем элемент по вычисленному смещению
   imul ecx, edi, M            ; Аналогично считаем смещение в массиве B
   add  ecx, esi
   mov  dword [B + 4*ecx], eax ; и записываем значение элемента

   inc  edi                    ; Проверка счетчика внутреннего цикла
   cmp  edi, N
   jl   .l2

   inc  esi                    ; Проверка счетчика внешнего цикла
   cmp  esi, M
   jl   .l1
```

Задача

Напишите фрагмент ассемблерной программы, в которой:

1) в статической памяти выделено пространство для матрицы 32-х разрядных целых чисел размером `N*N`,

2) выполняется поиск максимального по модулю элемента главной диагонали,

3) найденное число печатается на стандартный вывод.

Задача

Транспонируйте квадратную матрицу «на месте», не используя дополнительной памяти.