Семинар 26.02.2021

adc/sbb - сложение/вычитание с учётом CF

По заданному фрагменту на языке Си написать эквивалентный код на языке ассемблера.

```
static long long x;
…
x++;
```

```
section  .bss
   x resq 1
```

```
section  .text
   add  dword[x], 1     ; Сложение младших частей
   adc  dword[x + 4], 0 ; Учитываем возможный перенос из
                        ; старшего разряда
```

neg - изменение знака

```
neg r/m
```

Примеры:

```
neg eax
neg bx
neg byte[ecx]
```

mul - беззнаковое умножение

```
mul r/m ; результат = r/m * неявный операнд
```

Размер явного операнда	1	2	4
Неявный операнд	al	ax	eax
Результат	ax	dx:ax	edx:eax

Флаги: CF=OF=1, если переполнение размера неявного операнда, остальные не определены

Примеры:

```
mul ecx ; edx:eax = ecx * eax
mul word[a] ; dx:ax = word[a] * eax
mul al ; ax = al * al
```

imul - знаковое умножение

```
imul r/m ; как mul: результат = r/m * неявный операнд
imul r/m16/32, r/m/imm16/32 ; op1 *= op2
imul r16/32, r/m16/32, imm ; op1 = op2 * op3
```

Флаги и неявные операнды как mul

Примеры:

```
imul ebx ; edx:eax = ebx * eax
imul cx, word[edx] ; cx = cx * word[edx]
imul edx, dword[a], 100 ; edx = dword[a] * 100
```

Задача

Записать эквивалентную данному фрагменту на языке Си программу на языке ассемблера. Описать секции кода, инициализированных и неинициализированных данных.

```
static int a, b = 1, c = -2, d = 3;
...
a = b + c * d;
```

div/idiv - беззнаковое/знаковое деление и взятие остатка

```
div/idiv r/m ; частное = неявный операнд / r/m
; остаток = неявный операнд % r/m
```

Размер явного операнда	1	2	4
Неявный операнд	ax	dx:ax	edx:eax
Частное	al	ax	eax
Остаток	ah	dx	edx

Флаги не определены, при переполнении генерируется исключение

Примеры:

```
div cl ; al = ax / cl, ah = ax % cl
idiv word[a] ; ax = dx:ax / word[a], dx = dx:ax%word[a]
div ebx ; eax = edx:eax / ebx, edx = edx:eax % ebx
```

Расширение перед делением обязательно: mov, cbw, cwd, cdq

В случае беззнакового деления (`div`):

```
mov ah, 0 ; расширение al до ax
mov dx, 0 ; расширение ax до dx:ax
mov edx, 0 ; расширение eax до edx:eax
```

В случае знакового деления (`idiv`):

```
cbw ; расширение al до ax
cwd ; расширение ax до dx:ax
cdq ; расширение eax до edx:eax
```

Примеры:

```
mov eax, -10
cdq ; edx = 0xFFFFFFFF
mov eax, 10
cdq ; edx = 0x00000000
```

Задача

Записать эквивалентную данному фрагменту на языке Си программу на языке ассемблера.

```
static int x, y;
...
x /= -y;
```

inc/dec - инкремент/декремент

```
inc r/m ; r/m += 1
dec r/m ; r/m -= 1
```

Не обновляет флаг `CF`, остальные устанавливаются в соответствии с результатом

Примеры:

```
mov eax, 100
inc eax ; eax = 101
mov byte[a], 0
dec byte[a] ; byte[a] = -1, CF не изменён
```

Итог по флагам

Таблица 2. Коды описания флагов.

T	Значение флага влияет на выполнение инструкции
M	Инструкция меняет флаг (устанавливает или сбрасывает, в зависимости от операндов)
-	Влияние инструкции на флаг не определено
Пусто	Инструкция не влияет на флаг

Таблица 3. Перекрестные ссылки регистра флагов.

	OF	SF	ZF	PF	CF
ADC, SBB	M	M	M	M	TM
ADD, SUB, NEG	M	M	M	M	M
MUL, IMUL	M	-	-	-	M
DIV, IDIV	-	-	-	-	-
DEC, INC	M	M	M	M	
MOV, XCHG, MOVSX, MOVZX, LEA					

Задача

Пусть

```
static unsigned short n; // 100 <= n <= 999
```

Приведите фрагмент ассемблерного кода для записи в n числа, полученного выписыванием в обратном порядке десятичных цифр исходного числа n.

shl/shr/sal/sar - сдвиги

```
shl/shr r/m, imm8/cl (< 32) ; логический (заполнение 0) сдвиг влево/вправо
sal/sar r/m, imm8/cl (< 32) ; арифметический сдвиг (заполнение знаковым битом) влево/вправо
```

Последний выдвинутый бит записывается в `CF`

Можно использовать для быстрых умножений/делений на степени 2

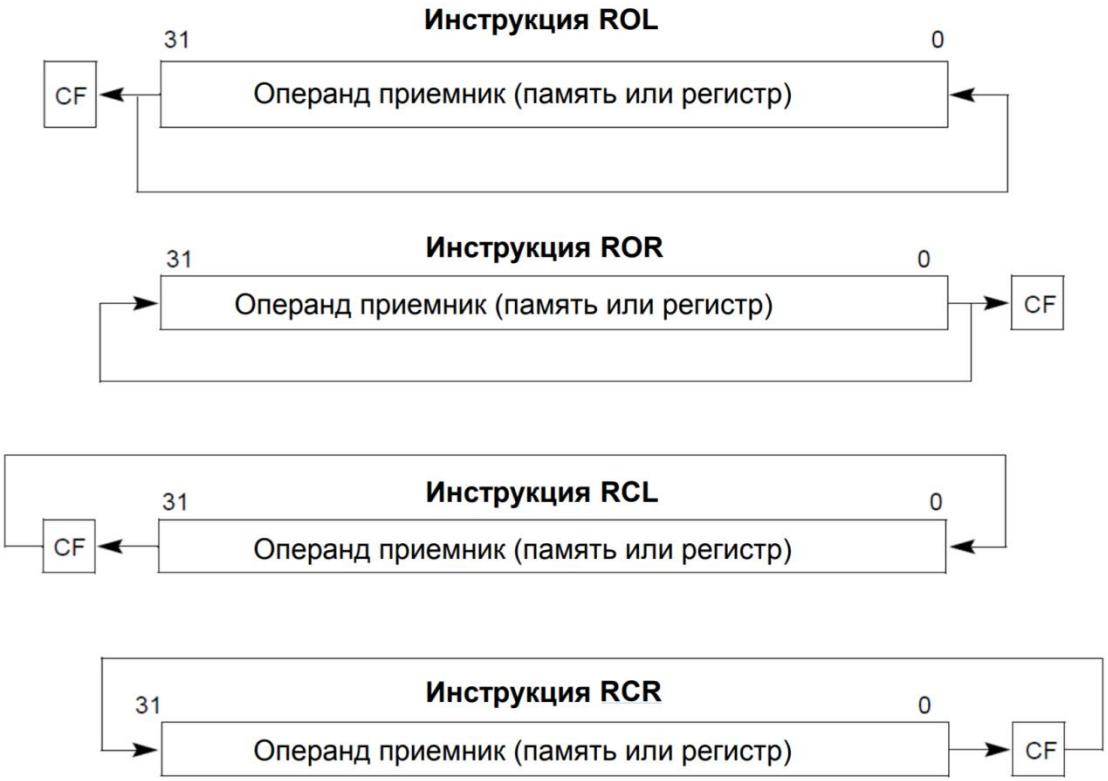
Примеры:

```
mov eax, 100
shl eax, 1 ; eax = eax * 2 = 200
mov edx, -64
sar edx, 2 ; edx = edx / 4 = -16
```

ror/rol/rcl/rcr - циклические сдвиги

`ror/rol/rcl/rcr r/m, imm8/cl (< 32)`

Последний выдвинутый бит записывается в `CF`



Задача

Приведите фрагмент ассемблерного кода для вычисления выражений.

```
// a
static unsigned x, y;
y = 32 * x - x / 8 + x % 16;

// b
static long long x;
x *= 2;

// c
static int x, y;
y = x / 64 + x * 4;
```

not/and(test)/xor/or - побитовые логические операции

```
not r/m
and(test)/xor/or r/m, r/m/imm
```

`not`: не меняет флаги

`and(test)/xor/or`: флаги `OF=CF=0`, остальные выставляются в соответствии с результатом

`test` аналогичен `and`, но не изменяет первый операнд - только выставляет флаги

Примеры:

```
mov eax, 16
or eax, 3 ; eax = eax | 3 = 10000b | 11b = 19
mov dword[ecx], -2 ; dword[ecx] = 11..10b
not dword[ecx] ; dword[ecx] = 1
xor eax, eax ; eax = 0
```

Задача

Даны четыре статические переменные целого типа a, b, c, d. Написать фрагмент кода на языке ассемблера, вычисляющий значение переменной a.

```
static int a, b, c, d;

a = ~(a & b) | ((~c & d) | (c & ~d));
```

bt/bts/btr/btc - манипуляции битами

```
bt r/m16/32, r16/32/imm8 ; test
bts r/m16/32, r16/32/imm8 ; set, бит op2 из op1 = 1
btr r/m16/32, r16/32/imm8 ; reset, бит op2 из op1 = 0
btc r/m16/32, r16/32/imm8 ; complement, бит op2 из op1 ^= 1
```

Устанавливает флаг `CF = бит op2 из op1`, остальные не определены

Примеры:

```
mov eax, 101b
bt eax, 0 ; CF = 1
bts eax, 1 ; CF = 0, eax = 111b
btr eax, 0 ; CF = 1, eax = 110b
btc eax, 2 ; CF = 1, eax = 010b
```