Команды Ассемблера (2)

Умножение и деление

Для правильного выполнения умножения и деления требуется повышенное внимание. Вы должны использовать только определенные регистры и иметь в виду, что результат и операнды записываются в два регистра – **(e)dx** и **(e)ax**

(либо нужно использовать расширенные команды*)

Правила:

Умножение

1. **Множимое** должно быть в ***eax, ax*** или ***al***
2. **Размер множимого** определяется по ***множителю***.
3. **Результат** – в ***edx: eax*** (если умножались двойные слова (32 бит)), ***dx: ax*** (если умножались слова (16 бит)) или ***ax*** (если умножались байты).

Умножение

команда	src (множитель)	множимое	результат
mul src (без знака)	регистр переменная	al ax eax	ax dx:ax edx:eax
imul src (со знаком)			

- Размер 2-го множителя (множимого) и результата определяется размером src
- CF = OF = 0, если старшая половина результата равна 0, CF = OF = 1, иначе

```
mov al, opByte
imul sourceByte         ; ax <-  al * sourceByte

mov ax, opWord
imul sourceWord         ; dx:ax <-  ax * sourceWord

mov eax, opDoubleWord
imul sourceDoubleWord ; edx:eax <- eax * sourceDoubleWord
```

Пример

```
a db 2
b dw 3
z dd ?
...
movzx ax, a (?)

imul b   ; dx:ax = a * b

; сохраняем результат
mov word ptr z, ax    ; младшая половина
mov word ptr z+2, dx  ; старшая половина
```

Умножение со знаком (расширенная команда-1)

команда	dst	src
imul dst, src	регистр	регистр переменная число

- Семантика: dst := dst * src
- Любые (!) регистры
- Возможно переполнение!
  - особое правило: CF = OF = 1, если переполнение

Пример - 1

```
; вычислить z = a * b * 3
a db -2
b dw 4
z dw ?   ; возможно переполнение
...
movsx bx, a

imul bx, b ; bx = a*b
imul bx, 3 ; bx = a*b*3

mov z, bx
```

Пример - 2

```
; вычислить z = a^2 + 3*a*b
a dw -2
b dw 3
z dd ?
...
movsx eax, a

movsx ebx, b
imul ebx, eax   ; ebx = a*b
imul ebx, 3     ; ebx = 3*a*b

imul eax, eax   ; eax = a^2

add eax, ebx    ; eax = a^2 + 3*a*b
mov z, eax
```

Умножение со знаком (расширенная команда-2)

команда	dst	src1	src2
imul dst, src1, src2	регистр	регистр переменная	регистр переменная число

- Семантика: dst := src1* src2
- Флаги и переполнение - как у **imul** dst, src

Пример - 3

```
; вычислить z = a^2 + 3*a*b
a dw -2
b dw 3
z dw ?     ; возможно переполнение
...
mov ax, a

imul bx, ax, b ; bx = a*b
imul bx, 3     ; bx = 3*a*b

imul ax, ax    ; ax = a^2
               ; или imul ax

add ax, bx     ; ax = a^2 + 3*a*b
mov z, ax
```

Деление

1. **Делимое** должно быть в **ax** (если делим на байт) или **(e)dx:(e)ax** (если делим на (двойное)слово).
2. Размер **делимого** определяется по ***делителю***. Делимое должно быть двойной длинны по отношению к делителю, для этого служат команды подготовки к делению **cbw, cwd, cwde, cdq, cdqe, cqo**.

3. **Результат** – в **(e)dx:(e)ax** (если делилось на (двойное) слово):
   (e)ax – частное, ***(e)dx*** – остаток от деления) или **ax** (если на байт: ***al*** – частное, ***ah*** - остаток). Любой остаток имеет тот же знак, что и частное.
4. Если вы попытаетесь выполнить деление на 0 или если результат от деления не будет помещаться в заданный операнд-назначение, то будет генерироваться **прерывание деления на 0**, останавливающее программу.

Подготовка к делению

команда	расшифровка	источник	приемник
cbw	convert byte -> word	al	ax
cwd	... word -> double word	ax	dx:ax
cwde	...	ax	eax
cdq	... double word -> quad word	eax	edx:eax
cdqe	...	eax	rax
cqo	quad word -> oct word	rax	rdx:rax

Деление

команда	src (делитель)	делимое	частное	остаток
div src (без знака)	регистр переменная	ax dx:ax edx:eax	al ax eax	ah dx edx
idiv src (со знаком)				

- Флаги OF, SF, ZF, AF, CF, PF не определены

Делим байт на байт

```
mov al, sourceByte ; исходный байт в al
cbw                ; Размножаем al в ax
idiv opByte        ; делим, рез-т в ax (частное – в al)
mov chastnoe, al
mov ostatok, ah
```

Делим слово на слово

```
mov ax, sourceWord  ; Загружает слово в ax
cwd                 ; Дублирует знак ax в dx
idiv opWord ; делим, рез-т в dx:ax (частное – в ax)
mov chastnoe, ax
mov ostatok, dx
```

Делим двойное слово на двойное слово

```
mov eax, sourceDWord  ; Загружает слово в eax
cdq                   ; Дублирует знак eax в edx
idiv opDWord ; делим, рез-т в edx:eax (частное – в
                      ;eax)
mov chastnoe, eax
mov ostatok, edx
```

Пример


```
; вычислить a/b, a%b
a dw -8
b dw 3
chastnoe dw ?
ostatok dw ?
...
mov ax, a
cwd
idiv b     ; ax = a/b,     dx = a%b

mov chastnoe, ax     ; чему равно?
mov ostatok, dx      ; чему равно?
```

Логические команды

Логические команды

and назначение, источник	Логическое И
not назначение	Логическое НЕ (отрицание)
or назначение, источник	Логическое ИЛИ
test назначение, источник	Проверка бита
xor назначение, источник	Логическое исключающее ИЛИ

Логические (побитовые) операции

команда	семантика	dst	src
and dst, src	И	регистр память	регистр память число
or dst, src	ИЛИ		
xor dst, src	исключающее ИЛИ		
test dst, src	"И" без сохранения результата		
not dst	НЕ		

Логические (побитовые) операции

- **not** не меняет флаги
- Все кроме **not**:
  - устанавливают флаги SF, ZF, PF
  - устанавливают OF = CF = 0
  - флаг AF не определен

Команды сдвига

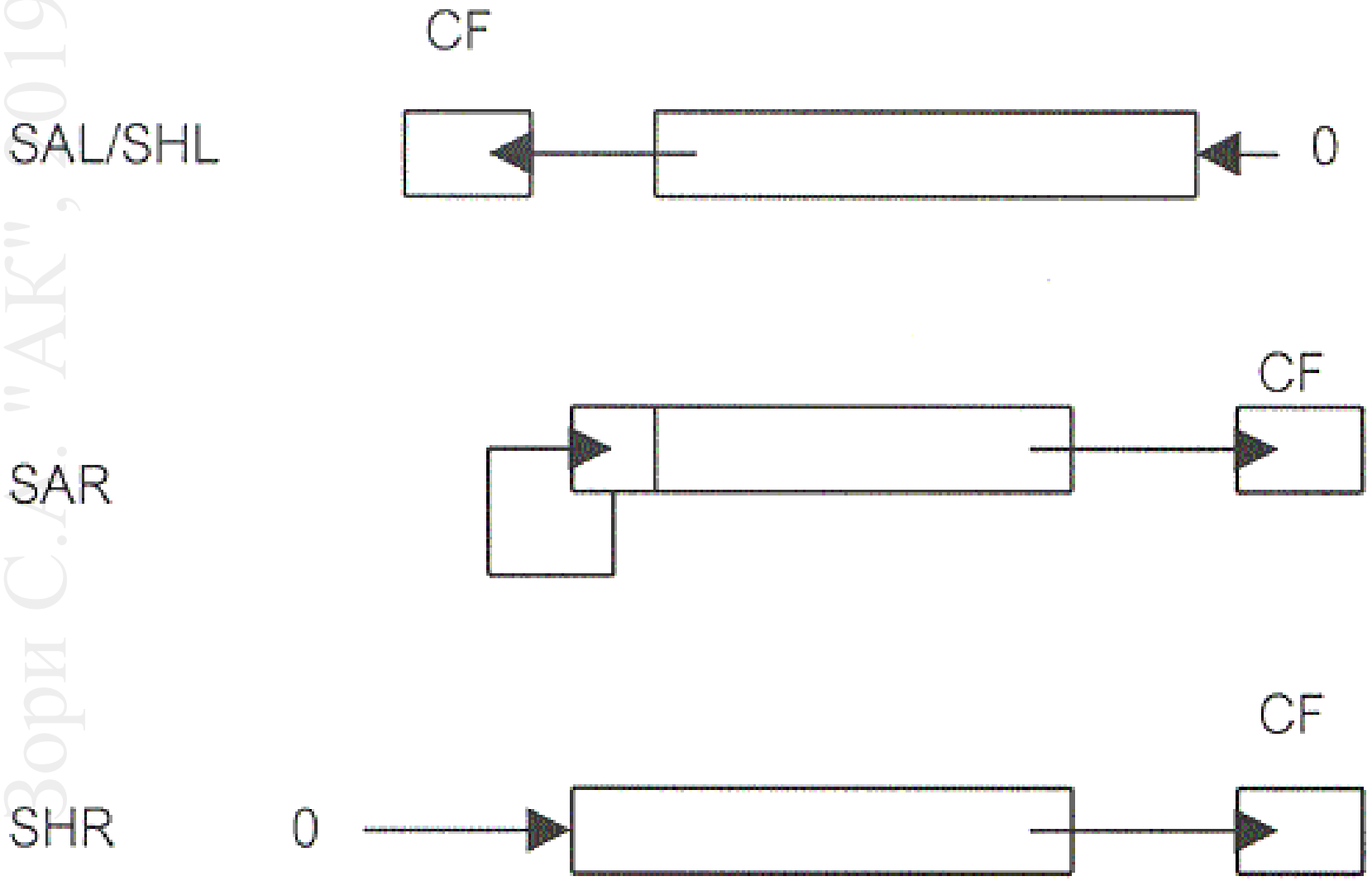
rcl назначение, количество	Циклический сдвиг влево через перенос
rcr назначение, количество	Циклический сдвиг вправо через перенос
rol назначение, количество	Циклический сдвиг влево
ror назначение, количество	Циклический сдвиг вправо
sar назначение, количество	Арифметический сдвиг вправо
sal назначение, количество	Арифметический сдвиг влево
shl назначение, количество	Сдвиг влево
shr назначение, количество	Сдвиг вправо

Сдвиговые операции

- Расшифоровка:
  - **sh** = **sh**ift
  - **sa** = **s**hift **a**rithmetic
  - **ro** = **ro**tation
  - **l** = **l**eft, **r** = **r**ight

- Флаги:
  - SF, ZF, PF
  - CF = вытолкнутый бит
  - AF не определен

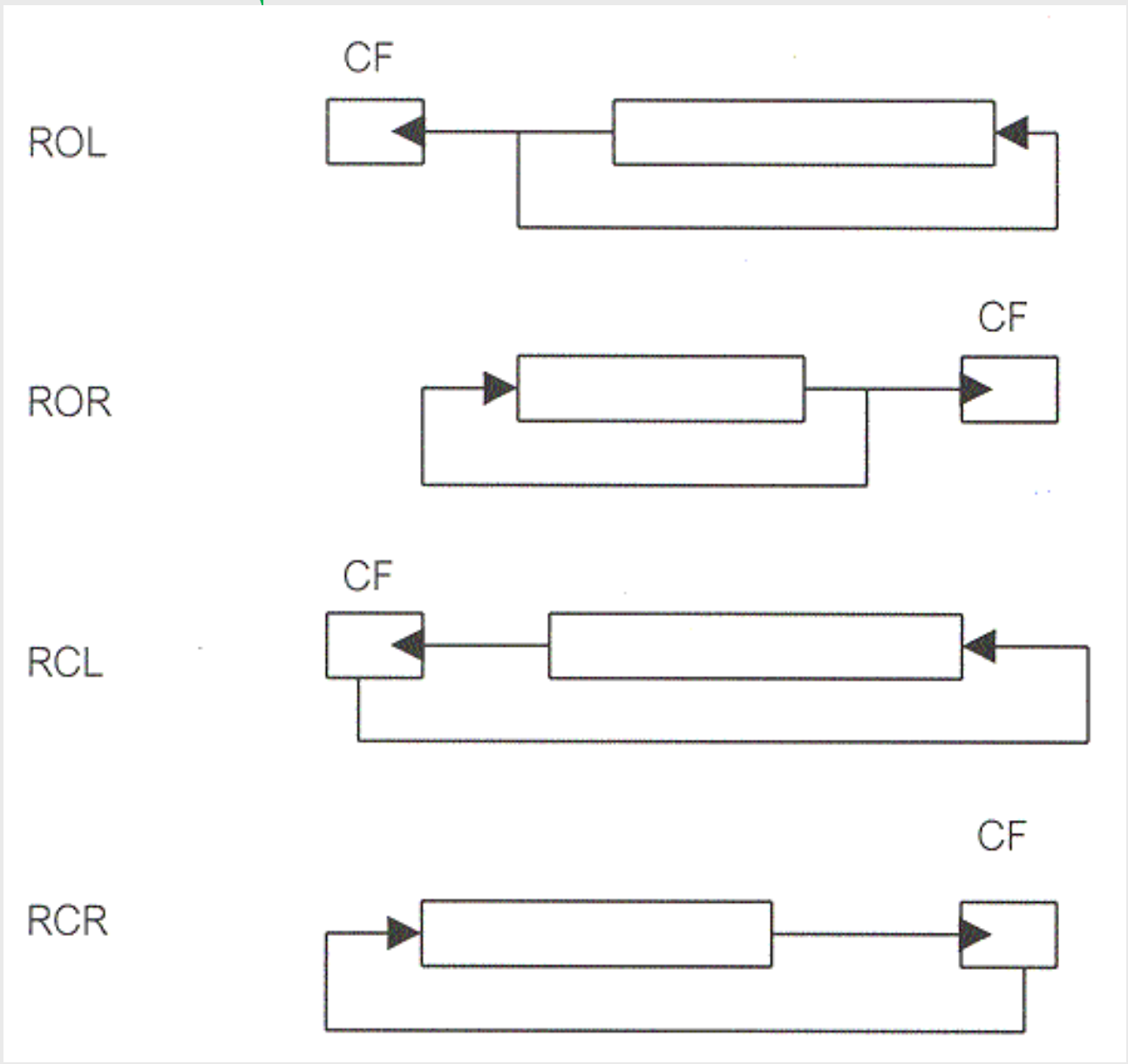
Сдвиговые операции



Сдвиговые операции

команда	сдвиг	флаг OF	dst	n
shr dst, n	логический вправо	старший бит dst до сдвига	регистр память	число, либо CL
sar dst, n	арифметический вправо	0		
shl dst, n **sal** dst, n	логический / арифметический влево	1, если старший бит изменился		

Циклический сдвиг



Циклический сдвиг

команда	сдвиг	dst	n
ror dst, n	вправо	регистр память	число, младшие 5-6 бит CL
rol dst, n	влево		
rcr dst, n	вправо через CF		
rcl dst, n	влево через CF		

- Применяются для хеширования, в криптографии
- Не реализованы в большинстве языков программирования

Сдвиг - либо на 1 бит:

```
shl ax, 1                ; Сдвиг ax влево на один бит
```

Либо на число бит, заданных в cl (cx):

сначала в регистр **cl** (*только cl!*) загружается счетчик сдвигов, затем cl используется в качестве второго операнда команды сдвига:

```
mov cl,5        ; Загрузить счетчик-сдвигов в cl
shl ax, cl      ; Сдвиг ax влево на 5 бит
```

Быстрая арифметика

- Битовые команды занимают меньше места, выполняются быстрее арифметических!

действие	**аналог**

dst *= 2^N	`mov cl,N`
	`sal dst,cl`
	`; или sal dst, N`

dst = 2^N	`mov dst, 1`
	`mov cl,N`
	`sal dst,cl`
	`; или sal dst, N`

Быстрая арифметика

действие	**аналог**

$dst = \lfloor log_2(src) \rfloor$	`mov dst, (число бит в src)-1`
	`bsr` `tmp, src`
	`sub dst, tmp`

$dst /= 2^N$ (без знака)	**`shr`** `dst, N`
$dst /= 2^N$ (со знаком)	**`sar`** `dst, N`
$dst \%= 2^N$ (без знака)	**`and`** `dst, 00..0`$\underbrace{11..1}_{N}$