

# БГТУ, ФИТ, ПОИТ, 3 семестр, Языки программирования

## Введение в язык Ассемблер

- 1. Регистр флагов EFLAGS:** флаги устанавливают инструкции процессора или специальные команды; непосредственно регистр не доступен программисту; программист может проверить состояние флагов (условные переходы).

**Флаг установлен**, когда значение соответствующего ему бита регистра EFLAGS равно 1.

**Флаг сброшен**, когда значение его бита регистра EFLAGS равно 0.

15							7							0	
0									0		0		1		
	NT	IOPL	OF	DF	IF	TF	SF	ZF		AF		PF		CF	
X	X	X	S	C	X	X	S	S		S		S		S	

Бит	Обозначение	Название	Описание
0	CF	Carry Flag	<b>Флаг переноса.</b> Устанавливается в 1, когда арифметическая операция генерирует перенос или выход за разрядную сетку результата. Сбрасывается в 0 в противном случае.
1	1		<b>Зарезервирован</b>
2	PF	Parity Flag	<b>Флаг чётности.</b> Устанавливается в 1, если результат последней операции имеет четное число единиц.
3	0		<b>Зарезервирован</b>
4	AF	Auxiliary Carry Flag	<b>Вспомогательный флаг переноса.</b> Устанавливается в 1, если арифметическая операция генерирует перенос из 3 бита в 4. Сбрасывается в 0 в противном случае. Этот флаг используется в двоично-десятичной арифметике.
5	0		<b>Зарезервирован</b>
6	ZF	Zero Flag	<b>Флаг нуля.</b> Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.
7	SF	Sign Flag	<b>Флаг знака.</b> Устанавливается равным старшему биту результата, который определяет знак в знаковых целочисленных операциях (0 – положительное число, 1 – отрицательное число).
8	TF	Trap Flag	<b>Флаг трассировки</b> (пошаговое выполнение).

9	IF	Interrupt Enable Flag	<b>Флаг разрешения прерываний.</b> При значении 1 микропроцессор реагирует на внешние аппаратные прерывания.
10	DF	Direction Flag	<b>Флаг направления.</b>
11	OF	Overflow Flag	<b>Флаг переполнения.</b> Устанавливается в 1, если целочисленный результат слишком длинный для размещения в целевом операнде. Этот флаг показывает наличие переполнения в знаковой целочисленной арифметике.
12 13	IOPL	I/O Privilege Level	<b>Уровень приоритета ввода-вывода.</b>
14	NT	Nested Task	<b>Флаг вложенности задач.</b>
15	0		<b>Зарезервирован</b>

## 2. Команды условного перехода:

Синтаксис команды условного перехода:

**Условие**

**метка\_перехода**

Команда условного перехода передает управление по указанной метке, если установлен соответствующий флаг состояния процессора. Если флаг сброшен, то выполняется следующая за ней команда.

## 2.1 Команды перехода в зависимости от значения флагов состояния процессора:

Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

Команда	Описание	Состояние флага
JZ	переход, если нуль	ZF=1
JNZ	переход, если не нуль	ZF=0

```
.code ; сегмент кода
main PROC ; начало процедуры
mov eax, 24
sub eax, 25 ;
jz zf1 ; if zf = 1 goto zf1
jnz zf0 ; if zf = 0 goto zf0
zf0:
mov ebx, 0
jmp fin
zf1:
mov ebx, 1
fin:
push 0 ; код возврата процесса (параметр ExitProcess)
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры

end main ; конец модуля, main - точка входа
```

Имя	Значение
ebx	0

```
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov eax, 24
sub eax, 24 ; zf = 1
jz zf1 ; if zf = 1 goto zf1
jnz zf0 ; if zf = 0 goto zf0
zf0:
mov ebx, 0
jmp fin
zf1:
mov ebx, 1
fin:
push 0 ; код возврата процесса (параметр ExitProcess)
call ExitProcess ; так должен заканчиваться любой процесс Windows
main ENDP ; конец процедуры

end main ; конец модуля, main - точка входа
```

Имя	Значение
ebx	1

```

ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov ax, 24
and ax, 11111100111b ; zf = 1
jz zf1 ; if zf = 1 goto zf1
jnz zf0 ; if zf = 0 goto zf0
zf0:
mov ebx, 0
jmp fin
zf1:
mov ebx, 1
fin:
push 0 ; код возврата процесса (п
call ExitProcess ; так должен заканчиваться
main ENDP ; конец процедуры

end main ; конец модуля, main - то

```

Имя	Значение
ebx	1
ax	0

## 2.2 Команды перехода в зависимости от значения флагов состояния процессора:

Флаг знака:

**SF (Sign Flag)** Устанавливается равным старшему биту результата, который определяет знак в знаковых целочисленных операциях (0 – положительное число, 1 – отрицательное число).

Команда	Описание	Состояние
JS	переход, если флаг знака установлен	SF=1
JNS	переход, если флаг знака сброшен	SF=0

```

ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov ax, 24
or ax, 1000000000000000b ; zs = 1
js zs1 ; if zs = 1 goto zs1
jns zs0 ; if zs = 0 goto zs0
zs0:
mov ebx, 0
jmp fin
zs1:
mov ebx, 1
fin:
push 0 ; код возврата процесса
call ExitProcess ; так должен заканчиваться
main ENDP ; конец процедуры

end main ; конец модуля, main - т

```

ebx	0x00000001
ax	0x8018

```

.code                                ; сегмент кода
main PROC                           ; начало процедуры
mov ax, 24
sub ax, 24                          ; zs = 0
jns zs1                             ; if zs = 1 goto zs1
jns zs0                             ; if zs = 0 goto zs0
zs0:
mov ebx, 0
jmp fin
zs1:
mov ebx, 1
fin:
push 0                             ; код возврата процесса
call ExitProcess                   ; так должен заканчива
main ENDP                           ; конец процедуры

end main                            ; конец модуля, main -

```

ebx	0x00000000
ax	0x0000

## 2.3 Команды перехода в зависимости от значения флагов состояния процессора:

Флаг чётности:

PF (Parity Flag) Устанавливается в 1, если результат последней операции имеет четное число единиц.

Команда	Описание	Состояние
JP	переход, если флаг четности установлен	PF=1
JNP	переход, если флаг четности сброшен	PF=0

```

.code                                ; сегмент кода
main PROC                           ; начало процедуры
mov ax, 0h
add ax, 3h                          ; pf = 1
jp pf1                              ; if pf = 1 goto pf1
jnp pf0                              ; if pf = 0 goto pf0
pf0:
mov ebx, 0
jmp fin
pf1:
mov ebx, 1
fin:
push 0                             ; код возврата процесса (пар
call ExitProcess                   ; так должен заканчиваться
main ENDP                           ; конец процедуры

end main                            ; конец модуля, main - точк

```

Имя	Значение
ebx	0x00000001
ax	0x0003

.const	; сегмент констант
.data	; сегмент данных
.code	; сегмент кода
main PROC	; начало процедуры
mov ax, 1h	
add ax, 3h	; pf = 1
jp pf1	; if pf = 1 goto pf1
jnp pf0	; if pf = 0 goto pf0
pf0:	
mov ebx, 0	
jmp fin	
pf1:	
mov ebx, 1	
fin:	
push 0	; код возврата процесса (пара
call ExitProcess	; так должен заканчиваться л
main ENDP	; конец процедуры
end main	; конец модуля, main - точка

Имя	Значение
ebx	0x00000000
ax	0x0004

## 2.4 Команды перехода в зависимости от значения флагов состояния процессора:

Флаг переполнения:

OF (Overflow Flag) Устанавливается в 1, если целочисленный результат слишком длинный для размещения в целевом операнде. Этот флаг показывает наличие переполнения в знаковой целочисленной арифметике.

Команда	Описание	Состояние
JO	переход, если возникло переполнение	OF=1
JNO	переход, если переполнения нет	OF =0

.data	; сегмент данных
.code	; сегмент кода
main PROC	; начало процедуры
mov al, 7fh	
add al, 1h	; of = 1
jo of1	; if of = 1 goto of1
jno of0	; if of = 0 goto of0
of0:	
mov ebx, 0	
jmp fin	
of1:	
mov ebx, 1	
fin:	
push 0	; код возврата процесса (пара
call ExitProcess	; так должен заканчиваться л
main ENDP	; конец процедуры
end main	; конец модуля, main - точка

Имя	Значение
ebx	0x00000001
al	0x80 'Б'

```

.code                                ; сегмент кода
main PROC                            ; начало процедуры
mov al, 7eh
add al, 1h                           ; of = 1
jg of1                                ; if of = 1 goto of1
jle of0                               ; if of = 0 goto of0
of0:
mov ebx, 0
jmp fin
of1:
mov ebx, 1
fin:
push 0                                ; код возврата процесса (параметр ExitProcess )
call ExitProcess                     ; так должен заканчиваться любой процесс Window
main ENDP                            ; конец процедуры

end main                             ; конец модуля, main - точка входа

```

Имя	Значение
ebx	0x00000000
al	0x7f 'f'

### 3. Команды сравнения

#### Команда TEST

Выполняет операцию поразрядного логического И между соответствующими парами битов двух операндов.

В зависимости от полученного результата устанавливает флаги состояния процессора.

Значение операнда-получателя не изменяется.

#### Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

```

.code                                ; сегмент кода
main PROC                            ; начало процедуры
mov al, 00001111b
test al, 00001000b                   ; and zf = 0
jz f1                                 ; if zf = 1 goto f1
jnz f0                                ; if zf = 0 goto f0
f0:
mov ebx, 0
jmp fin
f1:
mov ebx, 1
fin:
push 0                                ; код возврата процесса (параметр ExitProcess )
call ExitProcess                     ; так должен заканчиваться любой процесс Window
main ENDP                            ; конец процедуры

end main                             ; конец модуля, main - точка входа

```

Имя	Значение
ebx	0x00000000
al	0x0f '\xf'

```

include110 kernel32.lib ; компоновщик. компоновать
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov al, 00001111b
test al, 01100000b ; and zf = 1
jz f1 ; if zf = 1 goto f1
jnz f0 ; if zf = 0 goto f0
f0:
mov ebx, 0
jmp fin
f1:
mov ebx, 1
fin:
push 0 ; код возврата процесса (параметр)
call ExitProcess ; так должен заканчиваться процесс
main ENDP ; конец процедуры

end main ; конец модуля, main - точка входа

```

Имя	Значение
ebx	0x00000001
al	0x0f '\xf'

## Команда CMP

Команда вычитает исходный операнд из операнда получателя и устанавливает флаги – *флаг переноса* (CF), *флаг нуля* (ZF), *флаг знака* (SF), *флаг переполнения* (OF), *флаг четности* (PF), *флаг служебного переноса* (AF). Значение операнда-получателя не изменяется.

### Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой. Сбрасывается в 0 в противном случае.

```

ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov eax, 25h
mov ebx, 25h
cmp eax, ebx ; and zf = 1
jz f1 ; if zf = 1 goto f1
jnz f0 ; if zf = 0 goto f0
f0:
mov ebx, 0
jmp fin
f1:
mov ebx, 1
fin:
push 0 ; код возврата процесса
call ExitProcess ; так должен заканчиваться процесс
main ENDP ; конец процедуры

end main ; конец модуля, main - точка входа

```

ebx	0x00000001
al	0x25 '%'





## 4. Команды переходов при беззнаковом CMP-сравнении чисел CMP

4.1 Команды перехода в зависимости от равенства операндов или равенства нулю регистра ECX (CX)

**Флаг нуля:**

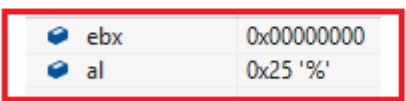
ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

Команда	Описание	Состояние
JE	переход, если равны	ZF=1
JNE	переход, если не равны	ZF=0

```
.model flat,stdcall      ; модель памяти, соглашения
includelib kernel32.lib  ; компоновщику: компоновать
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096              ; сегмент стека объемом 4096
.const                   ; сегмент констант
.data                   ; сегмент данных

.code                   ; сегмент кода
main PROC              ; начало процедуры
    mov eax, 25h
    mov ebx, 26h
    cmp eax, ebx       ; and zf = 1
    je fe              ; if eax == ebx goto fe
    jne fn             ; if eax != ebx goto fn
fe:
    mov ebx, 1
    jmp fin
fn:
    mov ebx, 0
fin:
    push 0              ; код возврата процесса (0 - успех)
    call ExitProcess    ; так должен заканчиваться процесс
    main ENDP           ; конец процедуры

end main                ; конец модуля, main - точка входа
```



## 4.2 Команды перехода в зависимости от равенства беззнаковых операндов

Команда	Описание	Состояние
JA	переход, если выше, т.е. левый операнд > правого операнда	ZF=1
JB	переход, если ниже, т.е. левый операнд < правого операнда	ZF=0

### Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

```

include lib kernel32.lib      ; компоновщику: компоновать
ExitProcess PROTO :DWORD      ; прототип функции
.stack 4096                    ; сегмент стека объемом 4096 байт
.const                         ; сегмент констант
.data                          ; сегмент данных

.code                           ; сегмент кода
main PROC                      ; начало процедуры
    mov eax, -200
    cmp eax, 100                ; and zf = 1
    ja fa                       ; if eax > 100 goto fa
    jb fb                       ; if eax < 100 goto fb
fa:
    mov ebx, 1
    jmp fin
fb:
    mov ebx, 0
fin:
    push 0                      ; код возврата процесса (0)
    call ExitProcess            ; так должен заканчиваться процесс
    main ENDP                  ; конец процедуры

```

Регистры (показаны в красной рамке):

ebx	0x00000001
al	0x38 '8'

### 4.3 Команды перехода в зависимости от равенства беззнаковых операндов

Команда	Описание	Состояние
JAE	переход, если выше, т.е. левый операнд $\geq$ правого операнда	ZF=1
JBE	переход, если ниже, т.е. левый операнд $\leq$ правого операнда	ZF=1

#### Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

```

.code                ; сегмент кода
main PROC           ; начало процедуры
mov  eax, -200
cmp  eax, -800      ; and  zf = 1
jae  fa             ; if  eax >= 100 goto fa
jbe  fb             ; if  eax <= 100 goto fb
fa:
mov  ebx, 1
jmp  fin
fb:
mov  ebx, 0
fin:
push  0              ; код возврата процесса (на
call  ExitProcess    ; так должен заканчиваться
main ENDP            ; конец процедуры

end main             ; конец модуля, main - точ

```

Имя	Значение
ebx	0x00000001
al	0x38 '8'
eax	0xffffffff38
-800	0xfffff3f0

## 5. Команды переходов при CMP-сравнении чисел со знаком

5.1 Команды перехода после выполнения команд сравнения операндов со знаком

Команда	Описание	Состояние
JG	переход, если больше, т.е. левый операнд > правого операнда	ZF=0
JL	переход, если меньше, т.е. левый операнд < правого операнда	ZF=0

**Флаг нуля:**

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

```
include lib kernel32.lib ; компоновщику: компоновать
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных

.code ; сегмент кода
main PROC ; начало процедуры
mov eax, -200
cmp eax, -800 ; and zf = 1
jg fa ; if eax > -800 goto fa
jl fb ; if eax < -800 goto fb
fa:
mov ebx, 1
jmp fin
fb:
mov ebx, 0
fin:
push 0 ; код возврата процесса (п
call ExitProcess ; так должен заканчиваться
main ENDP ; конец процедуры
```

Имя	Значение
ebx	0x00000001
al	0x38 '8'

## 5.2 Команды перехода после выполнения команд сравнения операндов со знаком

Команда	Описание	Состояние
JGE	переход, если больше или равно, т.е. левый операнд $\geq$ правого операнда	ZF=1
JLE	переход, если меньше или равно, т.е. левый операнд $\leq$ правого операнда	ZF=1

### Флаг нуля:

ZF (Zero Flag) Устанавливается в 1, если результат нулевой.  
Сбрасывается в 0 в противном случае.

```
ddx dd 800
.code
main PROC
mov eax, -200
cmp eax, ddx
jge fa
jle fb
fa:
mov ebx, 1
jmp fin
fb:
mov ebx, 0
fin:
push 0
call ExitProcess
main ENDP
end main
```

Имя	Значение
ebx	0x00000000
al	0x38 '8'

## 6. Пример программы сравнения двух строк

```
.stack 4096                ; сегмент стека объемом 4096
.const                     ; сегмент констант
.data                     ; сегмент данных
hw byte "Hello, World!!!"
pm byte "Привет, Мир!!!"

.code                     ; сегмент кода
main PROC                ; начало процедуры

    mov ecx, sizeof hw
    cmp ecx, sizeof pm
    je mje                ; if sizeof hw == sizeof pm
    ja mhw                ; if sizeof hw > sizeof pm

mpm:
    mov ebx, -1            ; hw < pm
    jmp fin

mje:                      ; sizeof hw == sizeof pm
    mov esi, 0
loopmje:
    mov al, hw[esi]
    cmp al, pm[esi]
    ja mhw
    jb mpm
    add esi, 1
    loop loopmje
    mov ebx, 0            ; hw = pm
    jmp fin

mhw:
    mov ebx, 1            ; hw > pm

fin:
    push 0                ; код возврата процесса (параметр)
    call ExitProcess       ; так должен заканчиваться любой процесс
main ENDP                ; конец процедуры

end main                  ; конец модуля, main - точка входа
```

## 7. Команды проверки и установки отдельных битов

Команды BT, BTR, BTC и BTC используются для работы с отдельными битами. Команды используют для организации семафоров.

Синтаксис команды тестирования бита:

**BT**

**строка\_битов, n**

### Флаг переноса:

CF (Carry Flag) Устанавливается в 1, когда арифметическая операция генерирует перенос или выход за разрядную сетку результата. Сбрасывается в 0 в противном случае.

Команда	Описание	Состояние
JC	переход, если перенос	CF=1
JNC	переход, если нет переноса	CF=0

```
include11b kernel32.lib ; компоновщик: компоновать с kernel32.lib
ExitProcess PROTO :DWORD ; прототип функции
.stack 4096 ; сегмент стека объемом 4096
.const ; сегмент констант
.data ; сегмент данных
; 876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code ; сегмент кода
main PROC ; начало процедуры
    bt b2, 7 ; cf = bt2[7]
    jc yes ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0 ; код возврата процесса (параметр ExitProcess )
    call ExitProcess ; так должен заканчиваться любой процесс Window
    main ENDP ; конец процедуры
    end main ; конец модуля, main - точка входа
```

ebx	0x00000000
b2	0x0004



## Команда тестирование бита:

```
.data                                ; сегмент данных
;                                876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code                                ; сегмент кода
main PROC                           ; начало процедуры
    bt b2, 2                        ; cf = bt2[2]
    jc yes                          ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0                          ; код возврата процесса (параметр ExitProcess )
    call ExitProcess                ; так должен заканчиваться любой процесс Windows
    main ENDP                       ; конец процедуры

    end main                        ; конец модуля, main - точка входа
```

## Команда тестирование бита с инверсией:

```
.stack 4096                         ; сегмент стека объемом 4096
.const                              ; сегмент констант
.data                                ; сегмент данных
;                                876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code                                ; сегмент кода
main PROC                           ; начало процедуры
    btc b2, 2                       ; cf = bt2[2] bt2[2] = !bt2[2]
    jc yes                           ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0                          ; код возврата процесса (параметр ExitProcess )
    call ExitProcess                ; так должен заканчиваться любой процесс Windows
    main ENDP                       ; конец процедуры

    end main                        ; конец модуля, main - точка входа
```

ebx	0x00000001
b2	0x0000

Команда тестирование бита с инверсией:

```

.data                                ; сегмент данных
;      876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code                                ; сегмент кода
main PROC                           ; начало процедуры
    btc b1, 2                       ; cf = bt1[2] bt1[2] = !bt1[2]
    jc yes                          ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0                          ; код возврата процесса (параметр ExitProcess )
    call ExitProcess                ; так должен заканчиваться любой процесс Windows
    main ENDP                       ; конец процедуры

end main                            ; конец модуля, main - точка входа

```

Имя	Значение
ebx	0x00000000
b1	0x0004

Команда тестирование бита с установкой:

```

.stack 4096                         ; сегмент стека объемом 4096
.const                              ; сегмент констант
.data                                ; сегмент данных
;      876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code                                ; сегмент кода
main PROC                           ; начало процедуры
    bts b1, 2                       ; cf = bt1[2] bt1[2] = 1
    jc yes                          ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0                          ; код возврата процесса (парам
    call ExitProcess                ; так должен заканчиваться лю
    main ENDP                       ; конец процедуры

end main                            ; конец модуля, main - точка

```

Имя	Значение
ebx	0x00000000
b1	0x0004

Команда тестирование бита со сбросом:

```

.data                                ; сегмент данных
;      876543210
b1 dw 0000000000000000b
b2 dw 0000000000000100b
.code                                ; сегмент кода
main PROC                           ; начало процедуры
    btr b2, 2                       ; cf = bt2[2] bt2[2] = 0
    jc yes                          ; cf == 1
    mov ebx, 0
    jmp fin
yes:
    mov ebx, 1
fin:
    push 0                          ; код возврата процесса (параметр Exit
    call ExitProcess                ; так должен заканчиваться любой проц
    main ENDP                       ; конец процедуры

end main                            ; конец модуля, main - точка входа

```

Имя	Значение
ebx	0x00000001
b2	0x0000