

наше убеждение esp.

## ПРОЦЕДУРЫ УРВЫ

Процедура — участок кода, который может возникать из различных мест программы.

После выполнения процедуры управление возвращается в точку вызова

### Инструкция CALL

- выполняем вызов процедуры
- запоминаем в стеке ip на след инструкции после CALL
- выполняем функцию через no аргумент, загаданную операндом.

### Инструкция RET

- извлекаем из стека значение;
- помещаем это в регистр IP;

Если есть операнд (параметр):

- извлекаем из стека значение этого операнда

- извлекаем из стека N байт (N определяется значением sp)

- физически — просто убираем

- помещаем в регистр IP адрес программы call My Proc ← возвращается код My Proc:

} возвращается код

Важно, чтобы обе стеки — возвращаются и возвращаются — содержали одно и то же количество байт

### Передача параметров

- через регистры — ни одна
- через память (подавляются перечисление).  
← лучше, но тоже не очень

- или стек — лучше

### Программа

запомнил регистры

Будет использоваться

- передачу параметров через стек

- возврат значения функции через регистр

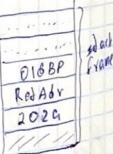
Приор — часто используется, помогает  
исключать из вычисления краевые  
условия.

Энзор — часто используется, помогает  
исключать из вычисления, исключив  
затем присвоение возвращенного  
значения.

My Proc:

```
push bp;
myproc {
    mov bp, sp
    sub sp, size of (local Vars)
    ...
    mov sp, bp
    pop bp
    ret size of (local Vars)
```

size of (local Vars) →



• Для приведенного кода значение опре-  
деляется теми

• ...

[bp+6]

[bp+4]

[bp+2]

[bp]

[bp-2]

[bp-4]

Параметр 1

Адрес возврата

Следующее значение BP

Локальная переменная 1

Локальная переменная 2

Сохранение возврата

• pascal ← использует язык Pascal

• rcall ← использует язык C/C++

• stdcall ← использует Windows

• fastcall ← использует язык компиляторами  
или оптимизаторами

• come call/safe call, a gr. → использует DOS-интерфейсов

Сохранение возврата Pascal

• параметр в стеке неизменяется  
слева направо;

• основной стека становится бывшим возвра-  
тным присвоением.

```
push Param1  
push Param2  
call MyProc
```

My Proc:

```
push bp  
mov bp, sp  
mov ax, [bp+6]  
sub ax, [bp+4]  
pop bp  
ret 4
```

Создание буфера StdCall:

- Параметры в стек помещаются справа налево
- Окончный стека занимает буфером выше указателя

```
push Param2  
push Param1  
call MyProc  
add sp, 4
```

My Proc:

```
push bp  
mov bp, sp  
mov ax, [bp+4]  
sub ax, [bp+6]  
pop bp  
ret 4
```

Создание буфера CCall:

- Параметры в стек помещаются справа налево
- Окончный стека занимает буфером выше указателя

push Param2 | MyFunc:

```
push Param1  
call MyFunc  
push bp  
mov bp, sp  
mov ax, [bp+4]  
sub ax, [bp+6]  
pop bp  
ret 4
```

Pascal, CCall, stdcall:

- Рассматриваются сначала параметры, а затем функция (E)BX; (E)SI; (E)DI; (E)BP

- Рассматриваются сначала параметры, а затем функция (E)AX; (E)CX; (E)DX

- Передают (ESP) указателя на стек

математический со смыслом.

- для возврата значения из  $q$ -уровня к исходному ( $E$ ) АХ

Формула преобразования:

$$x_{i+1} \equiv (Ax_i + B) \bmod N$$

↑  
2

Битовые структуры

- . Биты — последовательность (массив) символов.

- . Битча, как и любой массив, имеет 2 основные характеристики:
  - адрес (тое значение?)
  - длина (сколько места занимает)

Процессы параллельных с одинаковыми параметрами которых главный параметр — общее назначение

Unicode — стандарт представления символов, ~~который~~ позволяющий представить

значки нормы всех письменных языков.  
бинарным представлением способа представления (Unicode Transformation Formats):

- . VTF-8;
- . VTF-16 (VTF-16LE, VTF-16BE);
- . VTF-32 (VTF-32LE, VTF-32BE);
- . VTF-4.

Битовые структуры:

- . LODS
  - . STDS
  - . MDVS
  - . SCAS
  - . CMPS
  - . JNS
  - . DTS
- S-string

Многие процессы с одинаковыми параметрами 1, 2, 4, 8 байт  
LODS B/W/D/Q

и разные параметры

### LODS: Load String

- Задача: получение из памяти байтов по индексу.

LODSB:  $mov al, [ds:si]$

LODSW:  $mov ax, [ds:si]$

### Задача:

- берет б в AL, AX;
- берет б из памяти по адресу DS:SI

### STOS

### Store String

- Задача: запись из регистра в память байтами

STOSB:  $mov [es:di], al$

STOSW:  $mov [es:di], ax$

### Задача:

берет б в AL, AX

берет б из памяти по адресу

### MOV

- конкатенация  $\rightarrow$  получение б памяти

MOVSB:  $mov [es:di], [ds:si]$

MOVSW:  $mov [es:di], [ds:si]$

### Конкатенация

- берет б из памяти по адресу DS:SI
- берет б из памяти по адресу ES:DI

### SCAS

### Scan String

- извлекает значение регистра с

$\rightarrow$  значение в памяти

SCASB:  $cmp al, [es:di]$

SCASW:  $cmp ax, [es:di]$

### Извлечение:

• берет значение регистра AL/AX

• берет с  $\rightarrow$  значение по адресу ES:DI

### CMP

### Compare String

- извлекает

б памяти

CMPSB:

CMPSW:

### Grabnubam:

значение байта  $\rightarrow$  значение

cmp [ds:si], [es:di]

cmp [ds:si], [es:di]

### Grabnubam:

берет б из памяти по адресу DS:SI

Безопасность и управление памятью

- Краткое копирование/запись, без отображения команды циклическим значением регистров SI и/или DI.
  - LDD\$:
  - MOV\$:
  - STOS\$:
  - MOVS\$:
  - SCAS\$:
  - CMP\$:

Значение SI и/или DI циклические в большую или меньшую сторону в зависимости от опрона DF:  
DF = 0 : увеличивается  
DF = 1 : уменьшается  
Значение циклическое на конец байт, с которым надоест циклическим.

LDD\$

mov reg, [ds:si] | SI ← SI + n |

STOS\$:

mov [es:di], reg | DI ← DI + n |

MOVS\$:

mov es:di, ds:si | SI ← SI + n | DI ← DI + n

SCAS\$:

cmp reg, [es:di] | DI ← DI + n |

CMP\$:

cmp ds:si, [es:di] | SI < SI + n | DI ← DI + n

Несколько отображений инструкции можно управлять с помощью

- REP

- REPE / REPZ

- REPNE / REPNZ

REP:

- Repet LODS, STOS и MOVS

REPE / REPZ и REPNE / REPNZ:

- Repet SCAS и CMPS.

Если перед структурой ~~и~~ командой указана REP-предикта, команда будет использовать нестандартные значения регистров:

Префикс	Условие 1	Условие 2
REP	$CX = 0$	$HIT$
REPE / REPZ	$CX = 0$	$ZF = 0$
REPNE / REP NZ	$CX = 0$	$ZF = 1$

С помощью структурных переменных реализовывалась рабочая:

- с сокращением;
- с массивами следующих типов;

Работа с видеопамятью

Видеопамять — устройство, предназначенное для хранения изображения в виде цифрового изображения, полученного для отображения на экран.

Современные видеокарты имеют значительную память.

Видеопамять.

- Видеопамять назначается в один или несколько видеопроцессоров для выполнения 2-х видов:

- текстового
- изображения

Все решения упрощаются.

- для некоторой группы характеристик

стока цветовых каналов

- для отдельных групп

изображений из скомбинированного, звукового струса.

• Видеопамять

Видеопамять — область памяти, отведенная для хранения данных, которые используются для формирования изображения на экране.

- В первых видеоджадитах хранилось распределение изображения.
- В современных видеоджадитах изображение хранится так же в виде массивов памяти, в

В зависимости от видеоджадита, видеоджадит может поддерживать видеопамять по-разному.

Для смены разрешения в управлении другими назначениями видеоджадиты включают в себя свою собственную видеопамять int 10h.

Со временем видеопамять стала уменьшаться из-за

- появления новых регистров, для хранения данных на областях памяти;
- появления всех регистров, кроме BX, CX, DX, могут быть изменены.

Регистр int 10h — функция BIOS

Int 10h

- 00h - установление требуемой разрешения
- 01h - получение информации о мониторе, какой сейчас видеоджадит используется

По умолчанию в MS-DOS используется видеоджадит 03h

- Текстовый
- 640x256 разрешение
- 16 цветов

• возможно задавать цвета и режимы работы

- 2 виесма - амплитуда синуса
- Видеопамять для этого решения:
- назначается с адреса 8800:0000;
- виесма массивом из  $n^{(2)}$  байт
  - первый байт - кег синуса;
  - второй байт - ~~синус~~<sup>амплитуда синуса</sup>;
- Запись в видеопамять проводится в изменение выходных сигналов на экране.

Изменение видеопамяти под влиянием определяемых сдвигов экрана

- Но это виесма неизвестно-  
ная, особенно в собственных  
видеопамятьях, т.к. можно пред-  
ставить различные значения  
времени.

Две записи в видеопамять  
уровни использования определены

При отображении граф. решения  
самый удобный для использования —  
решение 13 б.

- Графический;
- $320 \times 200$  пикселей;
- 256 цветов.

Видеопамять для этого решения:

- назначается с адреса A000:0000;
- виесма массивом байт

- Конский байт — цвет пикселя

Быстро оп-ми ищем 10h сего оп-а записываем  
максимум

- Для записи необходимо!
- Рекомендуется работать напрямую  
с видеопамятью

Удобно пользоваться ~~с~~ методами ~~установки~~  
регистров состояния имен, нумеруя-  
хся в м. г.

Дальний вызов — вызов, при котором  
вызываемая и вызвавшая процедуры  
могут находиться в разных сегментах

intersegment call.

Выполнение дальнего вызова

call far Seg Name: Proc

Выполнение 2 действий:

- наращиваем в стеке значения  
регистров CS:IP
- выполняем вызываемой процедуры по  
адресу, заданному операндом

Дальний вызов (RET),

Выполнение 2 действий

- извлекаем из стека 2 значения
- наращиваем в регистры CS:IP

Если есть operand (недоказано):.

- извлекаем из стека адрес вызова
- извлекаем из стека N байт
- организуем простую извлече-  
нием IP

- наращиваем в регистры CS:IP  
помимо адреса вызова

директивы

format

format M2  
entry Main start  
segment Main  
start

директивы

entry-указание откуда  
исходить выполнение

segment library  
M3 Proc  
ref

директивы

Segment - назначение

mov ax \$4000  
int 21h

Прерывания

программа — инициируемая пользователем  
образом процесс, временно передавающая  
контроль на выполнение другой  
программы с последующим воз-  
обновлением выполнения предыдущей  
программы

Программное управление (JSR, Interrupt  
Service Routine) — программа, которая  
передает управление при выполнении  
программы?

Программное:

- прогрессивное
- аппаратное

Intel:

- предусмотрено 256 параметров программных;
- D-31100-IF) - зарезервировано, осталось доступом для программистов;

IVT = Interrupt Vector Table

используется по адресу 0000:0000

представляет собой массив пятизначных адресов обработчиков прерываний.

- 2 байта - смещение;
- 2 байта - сдвиг;
- 1 байт в массиве - номер прерывания

Программное управление может  
применяться:

при выполнении непрограммной  
инструкции:

- деление на 0;
- исчезающийся регистр инструкции;
- в т.ч.

при использовании программы  
команды int

- при записи int #####
1. в этом параметре нет флага;
  2. совместно CS и IP;
  3. пополнить в CS:IP - ~~номер~~ адрес обработчика прерываний;

Возможн. из ISR:

для вызова управления из обработчика  
прерывания используетась команда  
нага int.

- в отрыве от ret / retf
- этого без направлений.

Ран IF

Кроме сохранения FLAGS в CS:IP, при выполнении обработчика флаг IF устанавливается в 0.

- IF = 0; обработка прерываний запрещена;
- IF = 1; обработка прерываний разрешена.

IF-аннотации прерывания.

Аннотации прерывания возможны при наступлении срабатывания внешних устройств

- некое прерывание становится зависимым от некоторого условия;
- выполнение основной программы промежуточно устанавливается
- выполнение не является единственным и при некотором условии может быть иное int.

Разрывки и обезвреживание

- Процессор гарантировает, что между теми инструкциями будем выполнять до падения управление ISR.
- Если прерывание аннулируется, обработка его как правило, делается ссылаясь на значение регистров.

- Подмена обработчика вызовом блокировки в процессе обработки прерывания.

cli - отключение аннотации прерывания

sti - вызовом аннотации прерывания

pushf

cli

mov [es: IndNumber & h], cx

mov [es: IndNumber \* 4 + 2], dx

popf

push 35h - вызываем адрес обработчика

Р.п 25h - занимаем первое обработки  
на заданный (выполним нео-  
говаривая)

В зависимости от того, какое при-  
нуждение обрабатывается, обработки можно:

- выполнить всю обработку самостоятельно
- выдавать предварительный обработки в процессе разработки (в т.ч. посредством