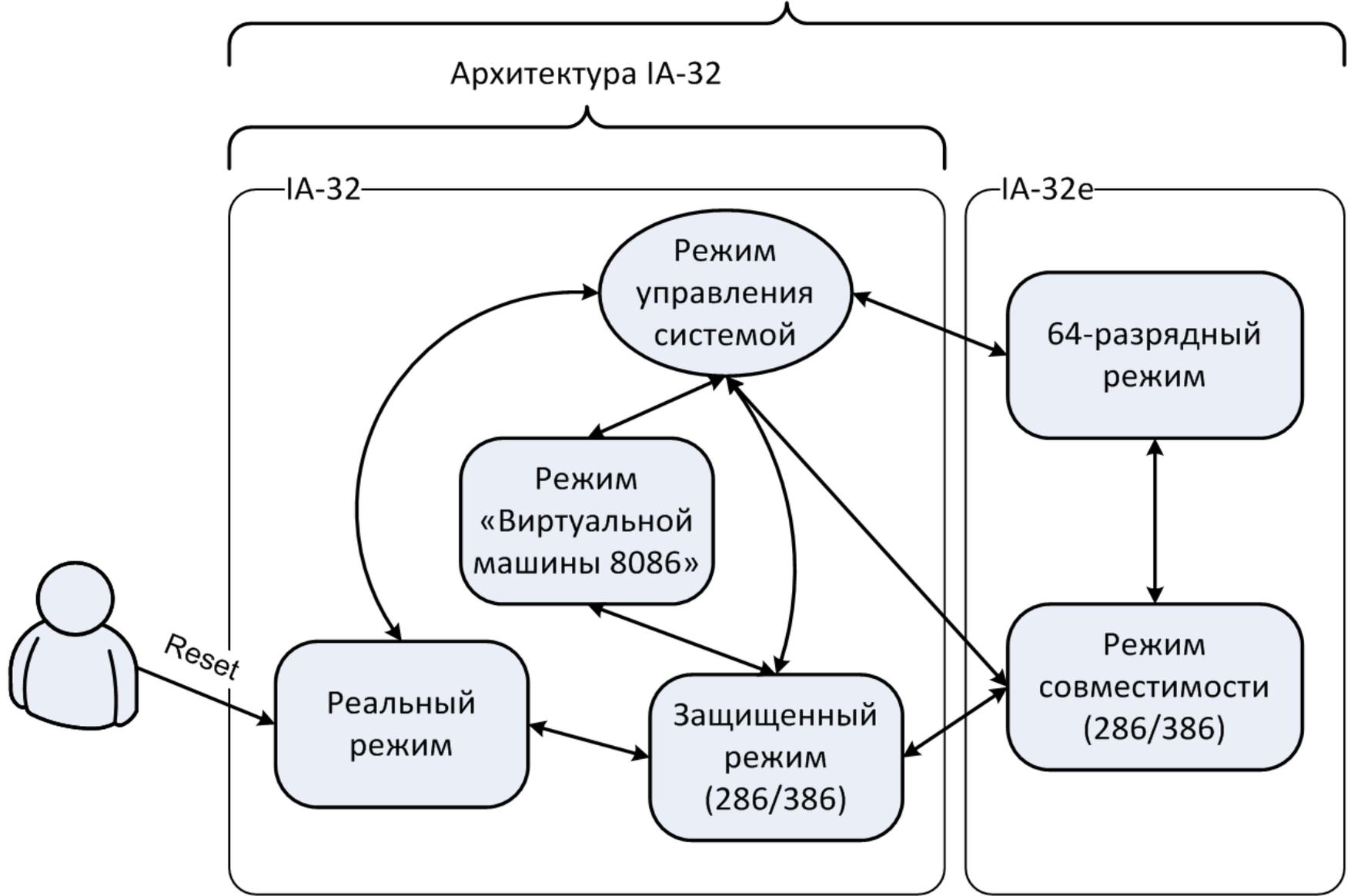


Лекция 22

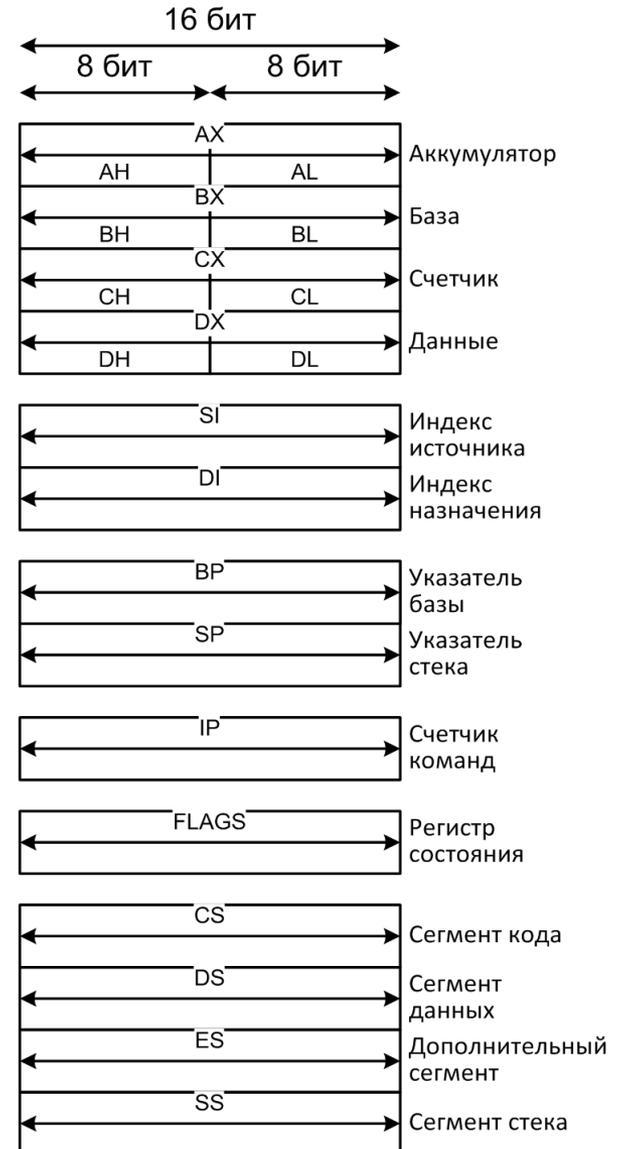
25 апреля

Архитектура Intel 64



Реальный режим / 8086

- Среда, в которой работает **одна** программа
 - Эта программа управляет всеми ресурсами
- Машинное слово 16 разрядов, адрес 20 разрядов
 - Сегментные регистры
 - Эффективный адрес = $(\text{seg_рег}) \ll 4 + \text{смещение}$
- Доступна вся память
- Периферийные устройства управляются через порты ввода/вывода



Обратная совместимость

- Обратная совместимость – свойство семейства процессоров. На более новом компьютере могут выполняться программы, рассчитанные на более ранние модели.
 - Прямая совместимость
- Линия A20. 8086 → 80286.
 - Контроллер клавиатуры Intel 8042
 - ; открываем адресную линию A20
 - in al, 92h
 - or al, 2
 - out 92h, al

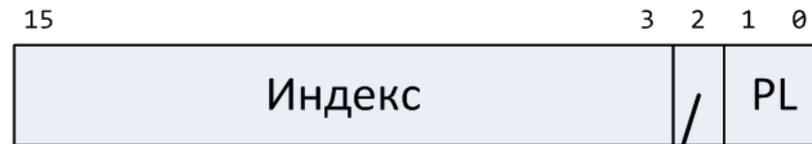
Многозадачная работа компьютера

- Привилегированный режим
 - Разделение машинных команд на две категории
- Механизм защиты памяти
- Таймер
- Механизм прерываний
 - Приоритеты прерываний
 - Одновременно произошедшие прерывания
 - Вложенные прерывания

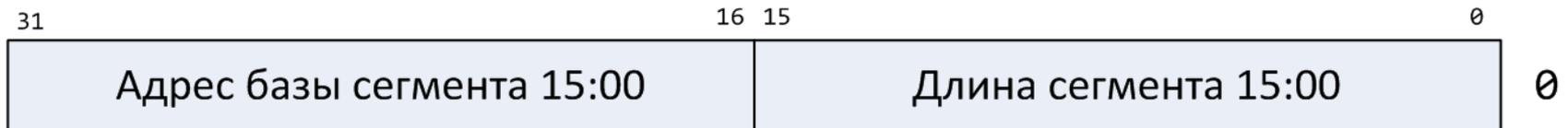
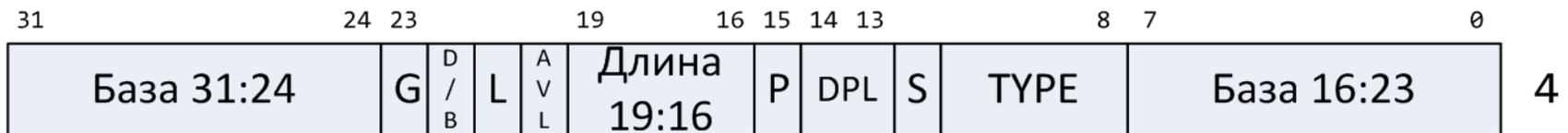
Защищенный режим / 80386

- Машинное слово и адрес 32 бита
- Два дополнительных сегментных регистра gs и fs
- Регистры управления
 - CR0, CR2, CR3
- Аппаратная защита памяти
 - Базы таблиц дескрипторов GDTR, LDTR, IDTR
 - Мультисегментная модель памяти
 - Фактически не используется
 - Многоуровневая трансляция адреса памяти
- Аппаратное переключение задач
 - фактически не используется

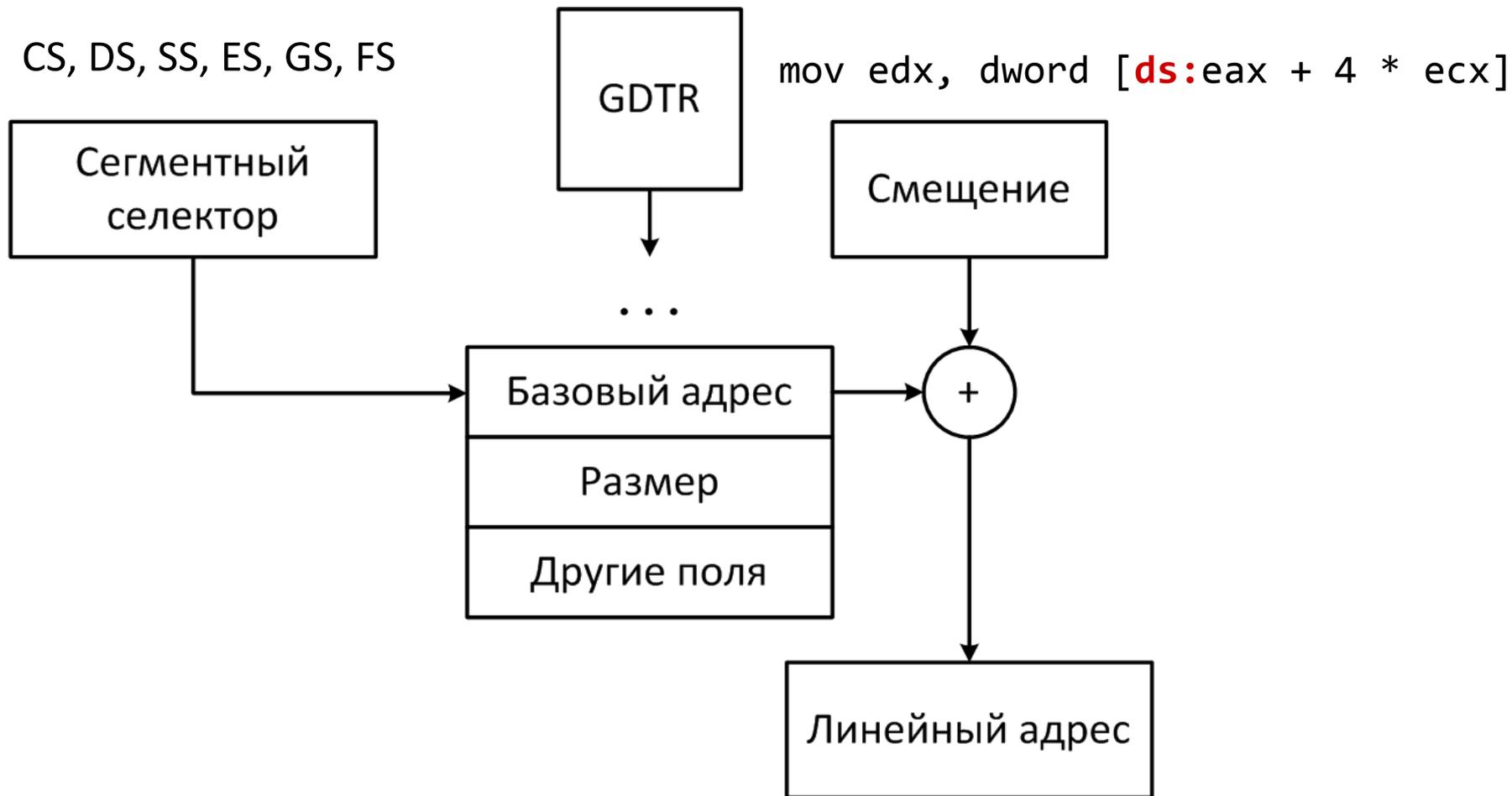
Сегментные селектор и дескриптор



0 = GDT
1 = LDT



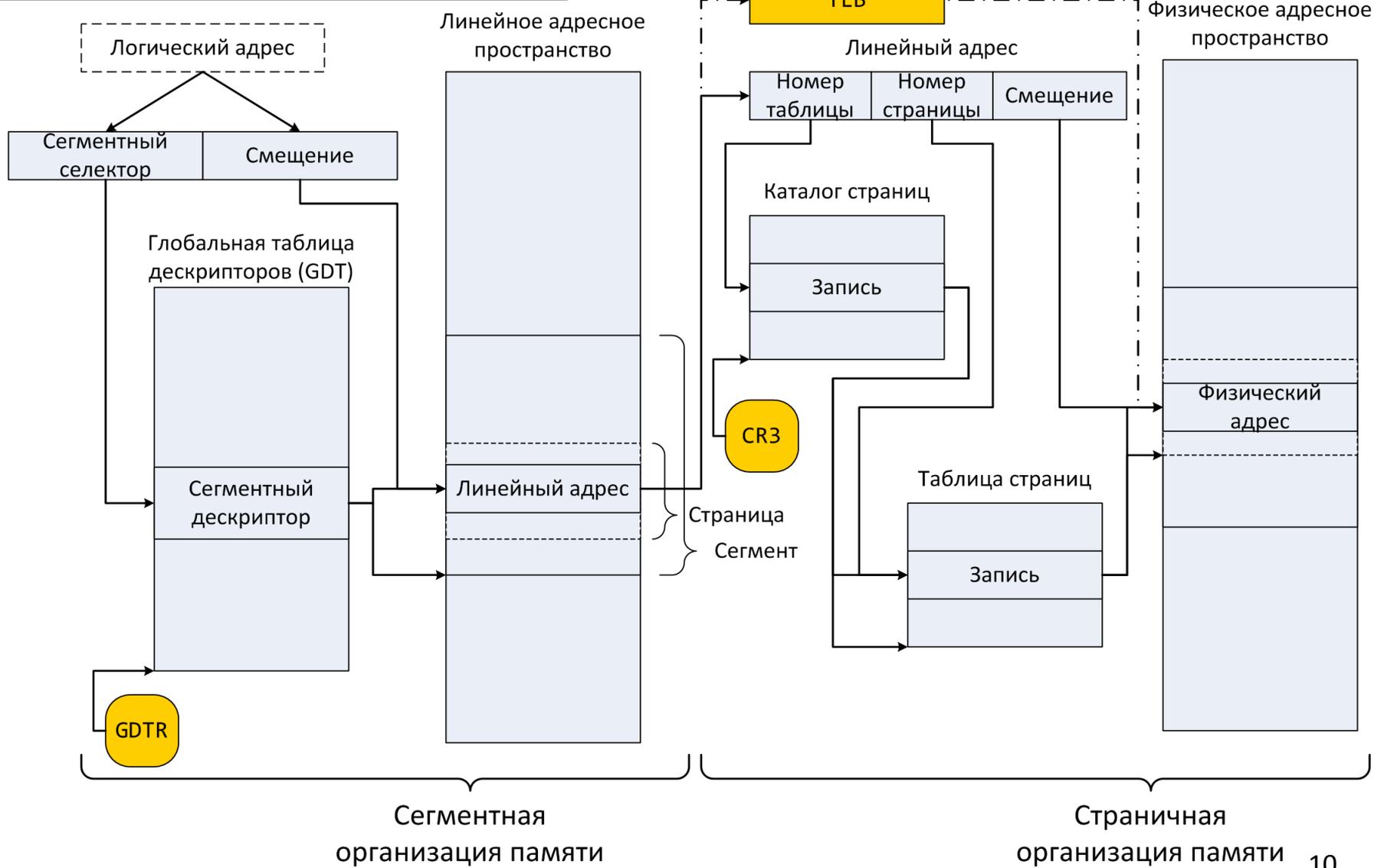
Адресация в защищенном режиме



Страничная организация памяти

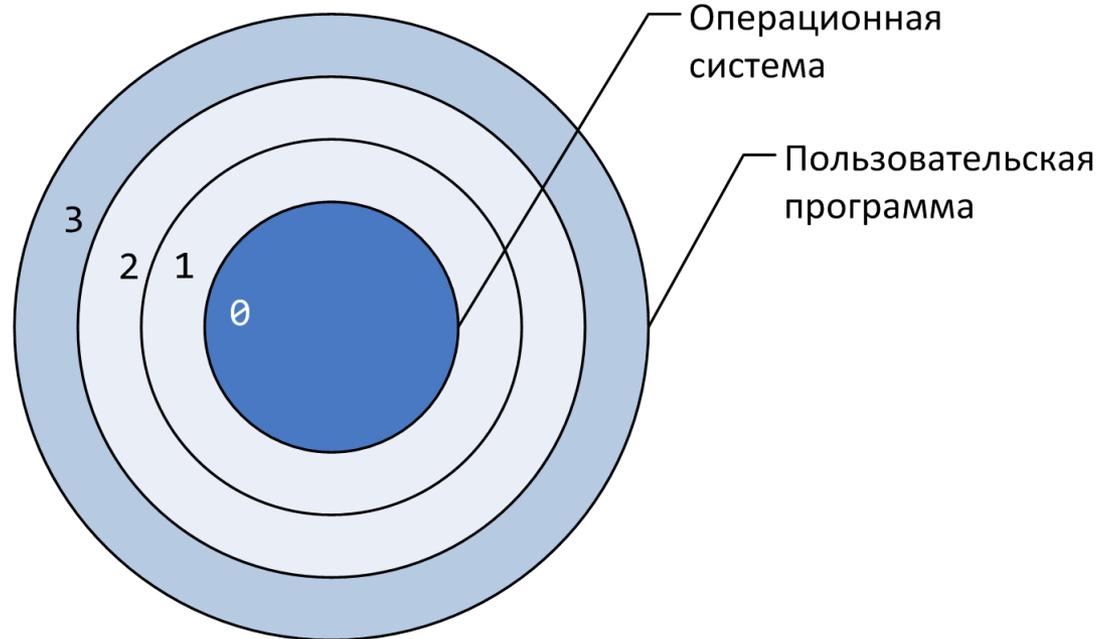
- Нехватка физической памяти
- Одновременная работа нескольких программ.
- Линейные адреса автоматически преобразуются в физические адреса.
 - Память разделена на фрагменты-страницы одинакового размера.
 - Старшие биты адреса меняются на биты, взятые из соответствующей записи таблицы страниц
- Отображение адресов выполняет ММУ

```
mov edx, dword [ds:eax + 4 * ecx]
```



Уровни защиты

- Каждый уровень привилегий обладает своим контекстом
- Стек вызовов – часть контекста
- Переход между уровнями контролируется полями PL в дескрипторах



Обращение к памяти: итоги

Канонический адрес

- Вычисляем смещение / адресный код в машинной команде
- Вычисляем линейный адрес / сегменты

Линейный адрес

- Пересчитываем линейный адрес в физический
 - Если есть запись в TLB берем готовый адрес
 - Если в TLB ничего нет, вычисляем физический адрес, проходя по таблицам трансляции

Физический адрес

- Обращаемся в память за данными
 - Берем из кэша, если данные в нем присутствуют

```
gdt:
gdt_null:
    dd    0, 0                ; Нулевой дескриптор
gdt_code:
    dw    0xFFFF            ; Сегмент кода
    dw    0
    db    0
    db    10011010b
    db    11001111b
    db    0
gdt_data:
    dw    0xFFFF            ; Сегмент данных
    dw    0
    db    0
    db    10010010b
    db    11001111b
    db    0
gdt_end:

; Значение, которое мы загрузим в GDTR:
gd_reg:
    dw    gdt_end - gdt - 1
    dd    gdt
```

```
[BITS 16]
```

```
[ORG 0x0700] ; начинаем располагать код с этого адреса
```

```
; инициализируем сегментные регистры
```

```
cli
```

```
mov ax, 0
```

```
mov ds, ax
```

```
mov es, ax
```

```
mov ss, ax
```

```
mov sp, 0x700
```

```
sti
```

```
; Конфигурируем контроллер прерываний
```

```
...
```

```
call enableA20
```

```
; Загрузка регистра GDTR:
```

```
xor ax, ax
```

```
mov ds, ax
```

```
lgdt [gd_reg]
```

```
; Установка бита PE регистра CR0
```

```
mov eax, cr0
```

```
or al, 1
```

```
mov cr0, eax
```

```
; Переходим в защищенный режим, в 32-бита
```

```
jmp 0x08:protected
```

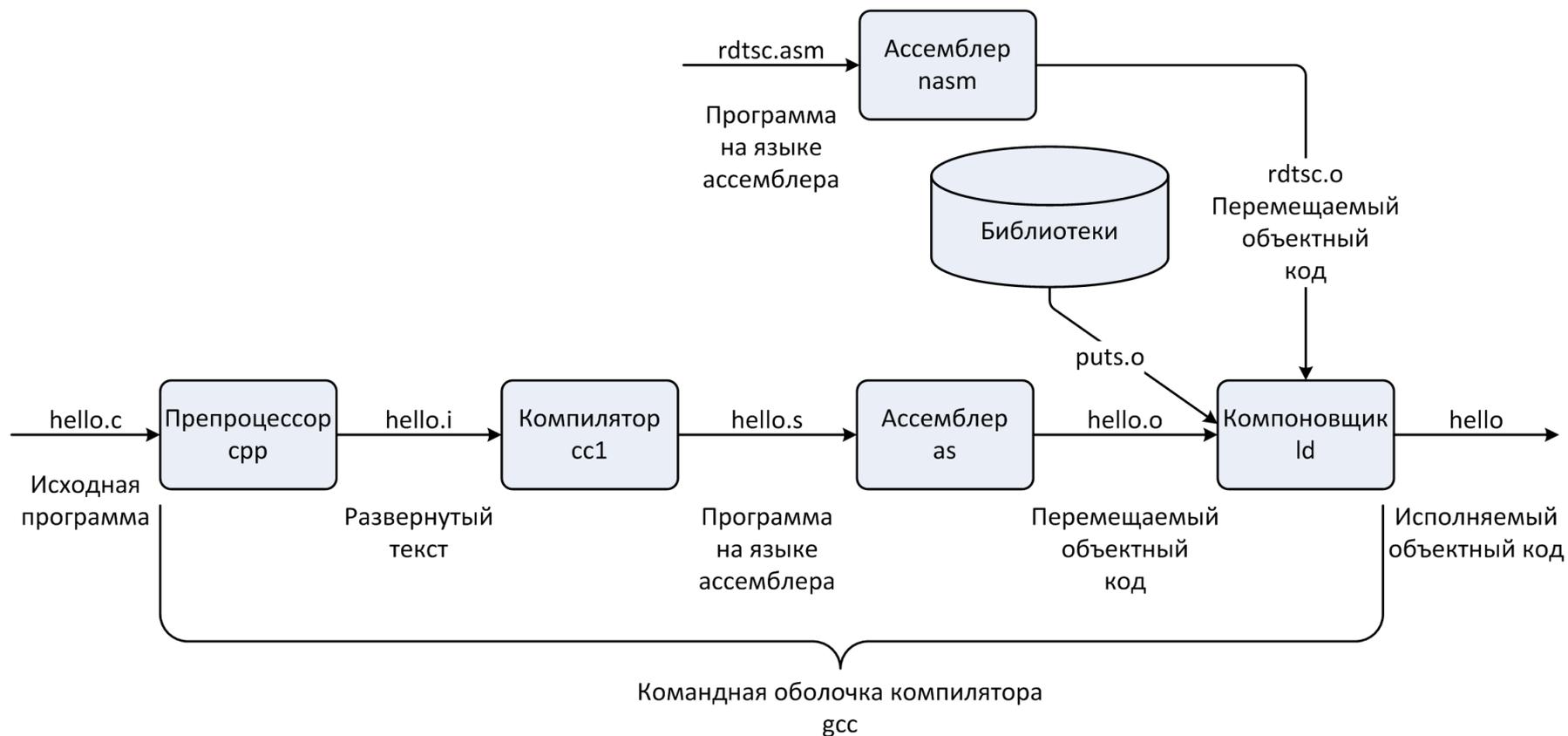
Язык программирования

- Цели, задачи, реализация языка
- Семантика
 - Операционная
- Компилируемые и интерпретируемые языки
 - Язык Java
исходный код → байт-код → машинные команды
- Процессор – интерпретатор машинных команд
 - Эффективность написания программы в машинных командах крайне низкая
 - Кодирование команд
 - Организация программы
 - Организация вычислений

Система программирования

- Системные/прикладные программы
 - Операционная система
 - Программные средства разработки
- Система программирования – комплекс средств
 - Язык программирования
 - Информационные ресурсы
 - Программные инструменты
 - Библиотеки
- Этапы жизненного цикла программы
 - Проектирование
 - Сбор и анализ требований к программе
 - Разработка
 - Реализация
 - Кодирование
 - Отладка
 - Сопровождение

Система программирования языка Си



Выполнение программы

