Организация памяти ЭВМ

Аппаратные средства вычислительной техники
ОГУ, кафедра ВТиЗИ
Галимов Р.Р. 2015

Список литературы

1. В.Ф. Мелехин. Вычислительные машины системы и сети.

Иерархия запоминающих устройств

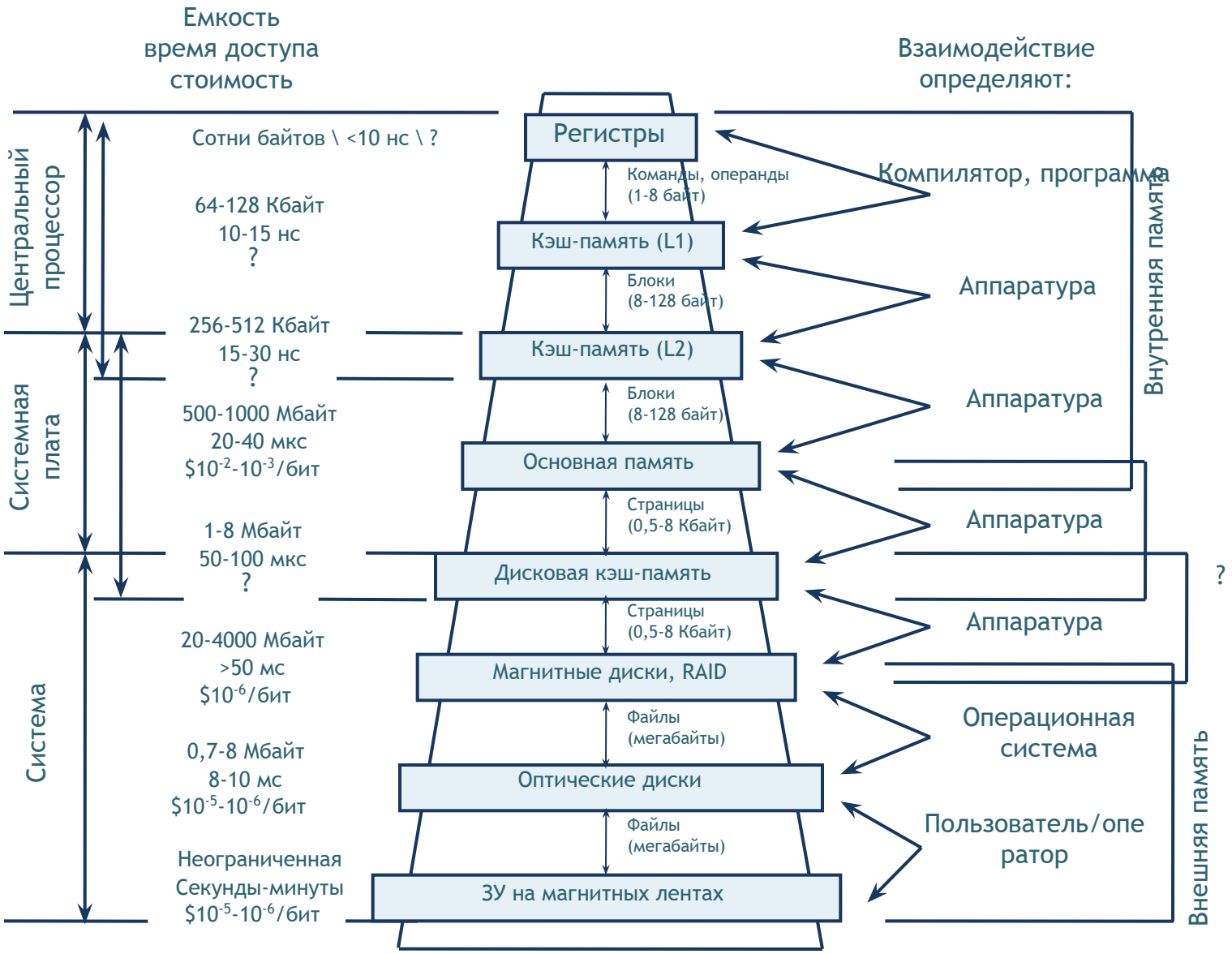
Память наряду с процессором в значительной мере определяет основные возможности ВМ — ее производительность и сложность решаемых задач, характеризуемую объемом программ и данных.

Основными системными требованиями, предъявляемыми к памяти являются:

- - большой информационный объем;
- малое время доступа к данным;
- низкая стоимость;
- энергонезависимость.

В настоящее время не существуют физические устройства памяти, полностью удовлетворяющие перечисленным системным требованиями, в связи с чем память ВМ реализуется не в виде отдельного устройства, а в виде иерархической многоуровневой системы, представляющей собой совокупность взаимодействующих устройств памяти.

Иерархия запоминающих устройств



Иерархия запоминающих устройств

Закономерности:

• чем меньше время доступа, тем выше стоимость хранения бита;

• чем больше емкость, тем ниже стоимость хранения бита, но больше время доступа.

При создании системы памяти постоянно приходится решать задачу обеспечения требуемой емкости и высокого быстродействия за приемлемую цену.

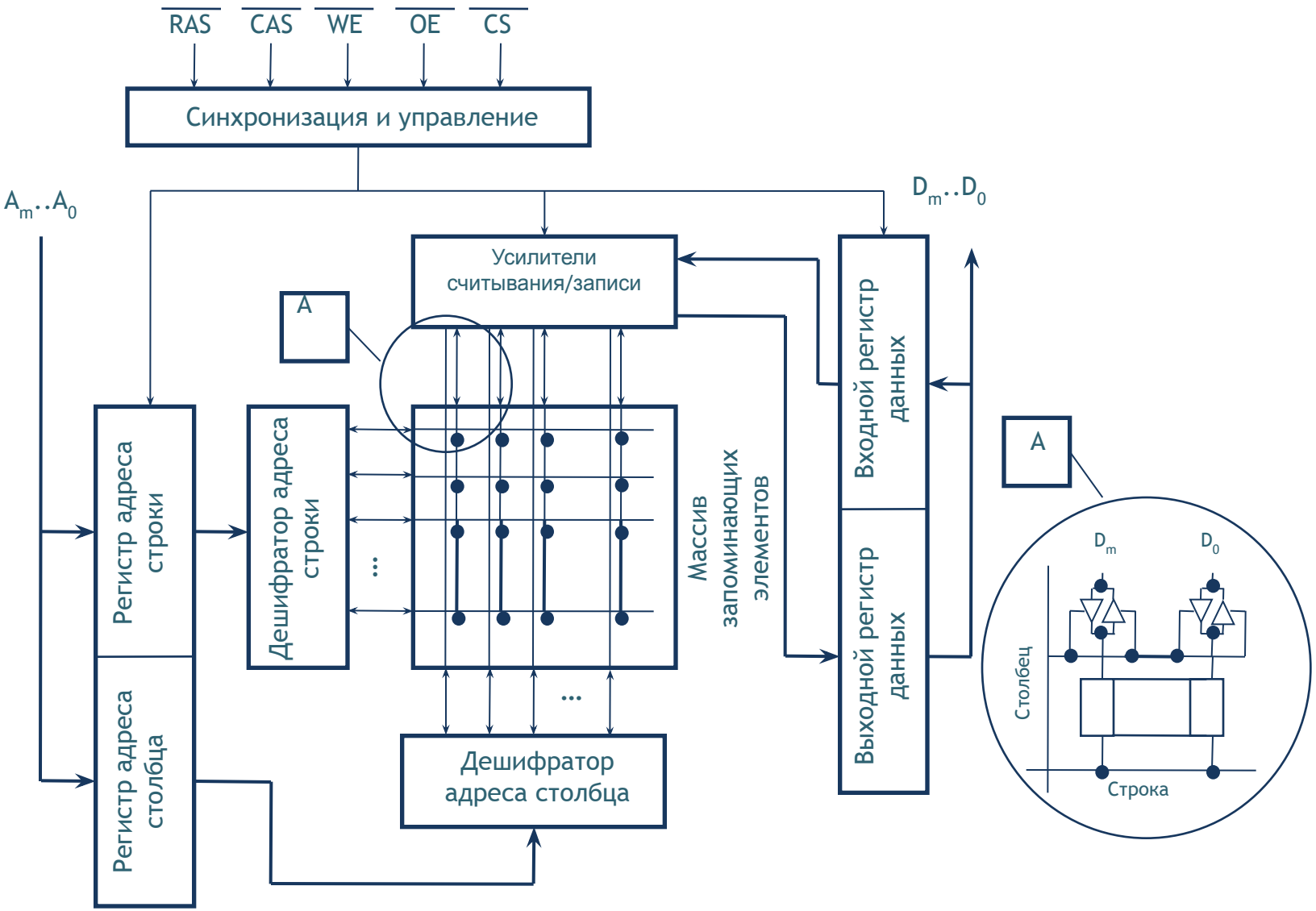
Уровни иерархии взаимосвязаны: все данные на одном уровне могут быть также найдены на более низком уровне, и все данные на этом более низком уровне могут быть найдены на следующем нижележащем уровне и т. д.

Иерархия запоминающих устройств

По мере движения вниз по иерархической структуре:

1. Уменьшается соотношение «стоимость/бит».
2. Возрастает емкость.
3. Растет время доступа.
4. Уменьшается частота обращения к памяти со стороны центрального процессора

Структура микросхемы памяти



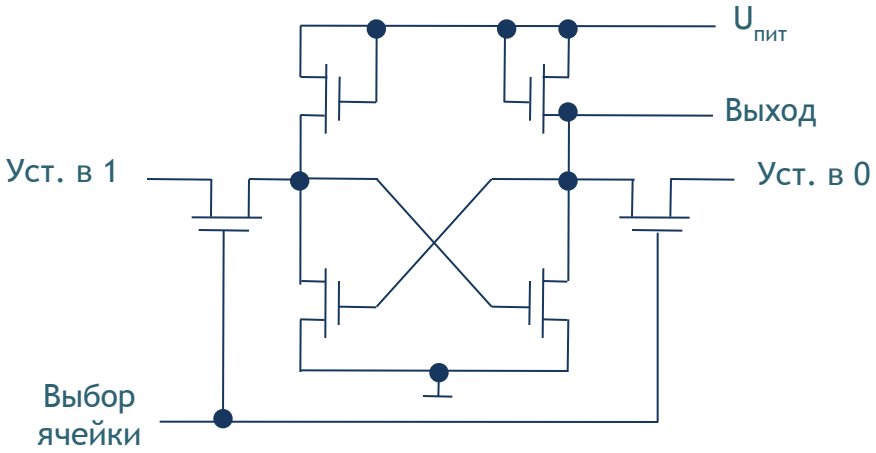
к содержанию

Статическая и динамическая память

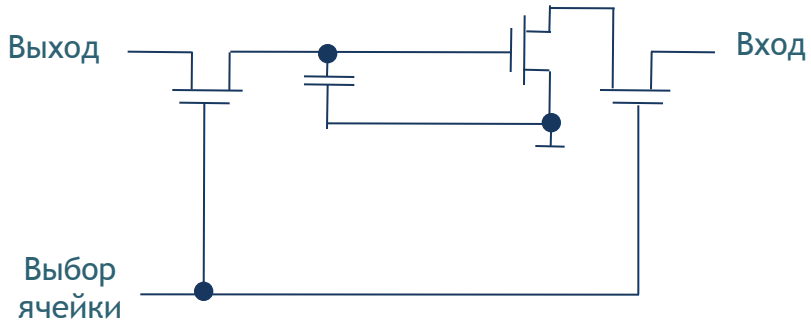
- Оперативная память может составляться из микросхем динамического (Dynamic Random Access Memory -DRAM) или статического (Static Random Access Memory -SRAM) типа.
- Память статического типа обладает более высоким быстродействием, но значительно дороже DRAM. В статической памяти элементы (ячейки) построены на различных вариантах триггеров – схем с двумя устойчивыми состояниями. При записи бита информации подобная ячейка может пребывать в данном состоянии долго при наличии питания.
- Ячейки SRAM , имеют малое время чтения/записи, но микросхемы на их базе отличаются низкой удельной емкостью и высоким энергопотреблением. Статическая память в основном используется как микропроцессорная память или кэш-память.

Запоминающие элементы

Запоминающий элемент статического ОЗУ



Запоминающий элемент динамического ОЗУ



к содержанию

Кэш-память

Кэш-память представляет собой быстродействующее ЗУ, размещенное на одном кристалле с ЦП или внешнее по отношению к ЦП.

Кэш служит высокоскоростным буфером между ЦП и относительно медленной основной памятью.

Идея кэш-памяти основана на прогнозировании наиболее вероятных обращений ЦП к *оперативной памяти*.

В основу такого подхода положен ***принцип временной и пространственной локальности программы.***

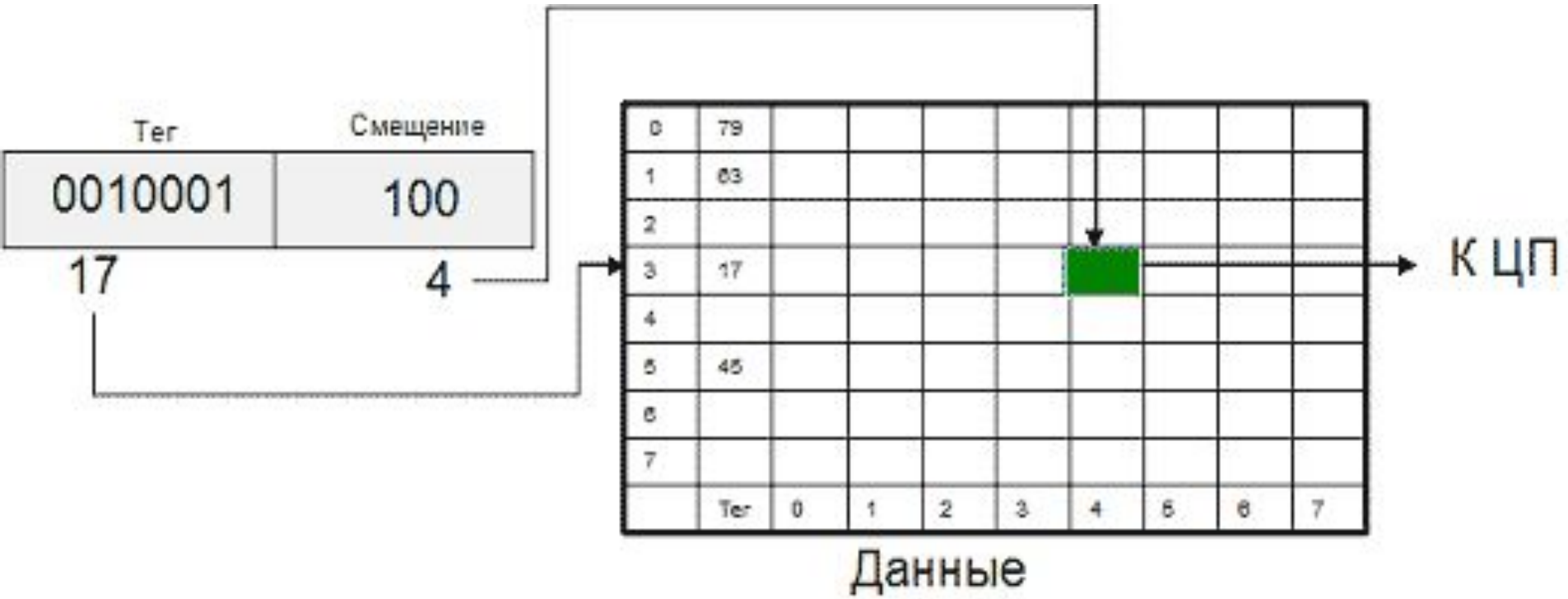
Кэш-память

В структуре кэш-памяти выделяют два типа блоков данных:

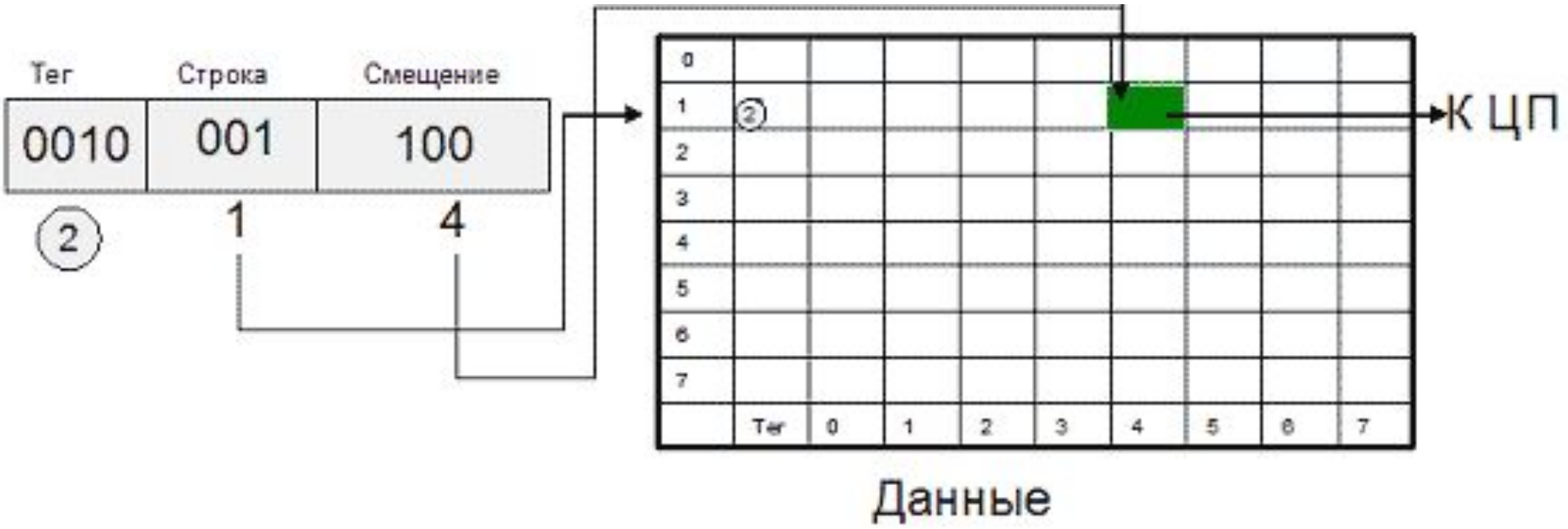
- **память отображения данных** (собственно сами данные, дублированные из *оперативной памяти*);
- **память тегов** (признаки, указывающие на расположение кэшированных данных в *оперативной памяти*).

По алгоритмам отображения *оперативной памяти* в *кэш* выделяют три типа кэш-памяти:

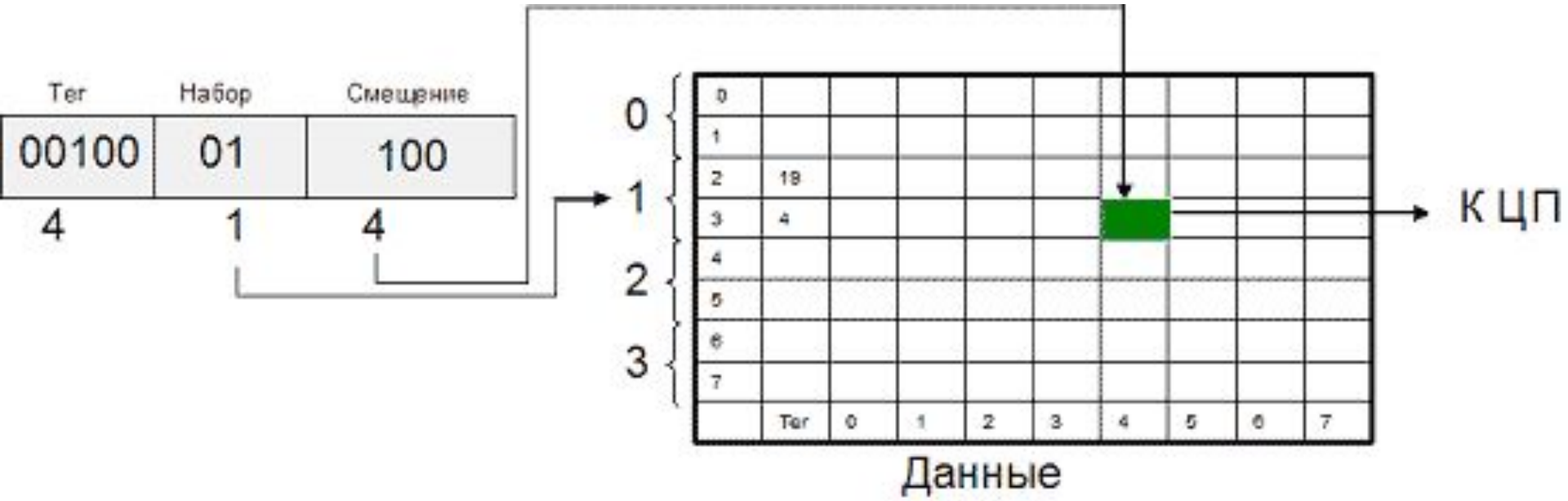
- полностью ассоциативный *кэш*;
- *кэш* прямого отображения;
- множественный ассоциативный *кэш*.



Полностью ассоциативный кэш 8x8 для 10-битного адреса



Кэш прямого отображения 8x8 для 10-битного адреса



Двухвходовый ассоциативный кэш 8x8 для 10-битного адреса

Для согласования содержимого кэш-памяти и *оперативной памяти* используют три метода записи:

- **сквозная запись** (write through) - одновременно с кэш-памятью обновляется *оперативная память*;

- **буферизованная сквозная запись** (buffered write through) - информация задерживается в кэш-буфере перед записью в *оперативную память* и переписывается в *оперативную память* в те циклы, когда ЦП к ней не обращается;

- **обратная запись** (write back) - используется бит изменения в поле тега, и строка переписывается в *оперативную память* только в том случае, если бит изменения равен 1.

	Intel486	Pentium	Pentium MMX	P6	Pentium 4
L1 кэш команд					
Тип	4-вх. ассоц.	2-вх. ассоц.	4-вх. ассоц.	4-вх. ассоц.	8-вх. ассоц.
Размер строки, байт	16	32	32	32	-
Общий объем, Кбайт	8/16	8	16	8/16	12Kmops
L1 кэш данных					
Тип	Общий с кэш инструкций	2-вх. ассоц.	4-вх. ассоц.	2/4-вх. ассоц.	4-вх. ассоц.
Размер строки, байт		32	32	32	64
Общий объем, Кбайт		8	16	8/16	8
L2 кэш					
Тип	Внешний	внешний 4-вх. ассоц.		4-вх. ассоц.	8-вх. ассоц.
Размер строки, байт		32		32	64
Общий объем, Кбайт		256/512		128-2048	256/512

$$T_{cp} = (T_{hit} \times R_{hit}) + (T_{miss} \times (1 - R_{hit}))$$

где T_{hit} - время доступа к кэш-памяти в случае попадания (включает время на идентификацию промаха или попадания), T_{miss} - время, необходимое на загрузку блока из основной памяти в строку *кэша* в случае кэш-промаха и последующую доставку запрошенных данных в процессор, R_{hit} - частота попаданий.

Структура стека

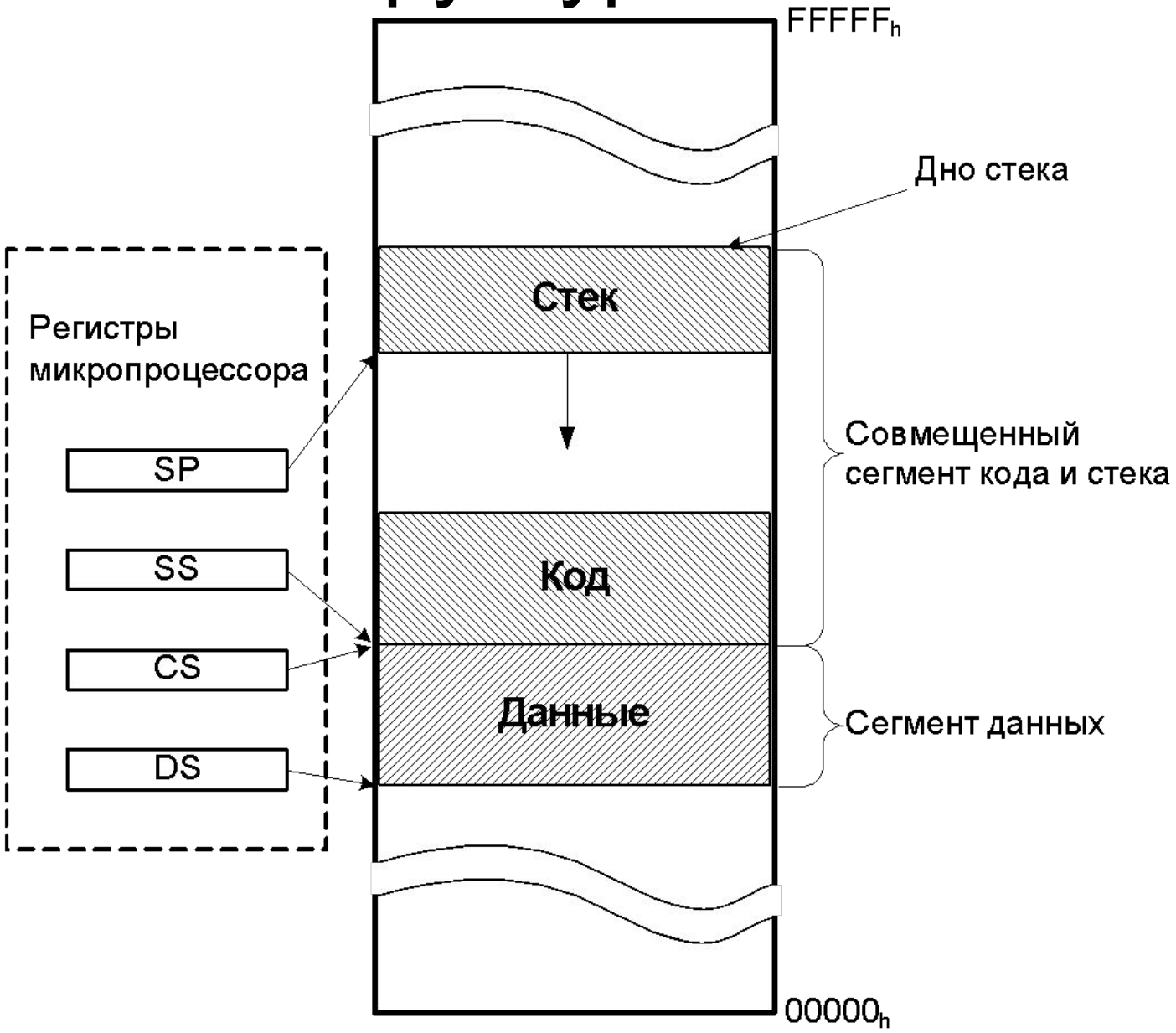
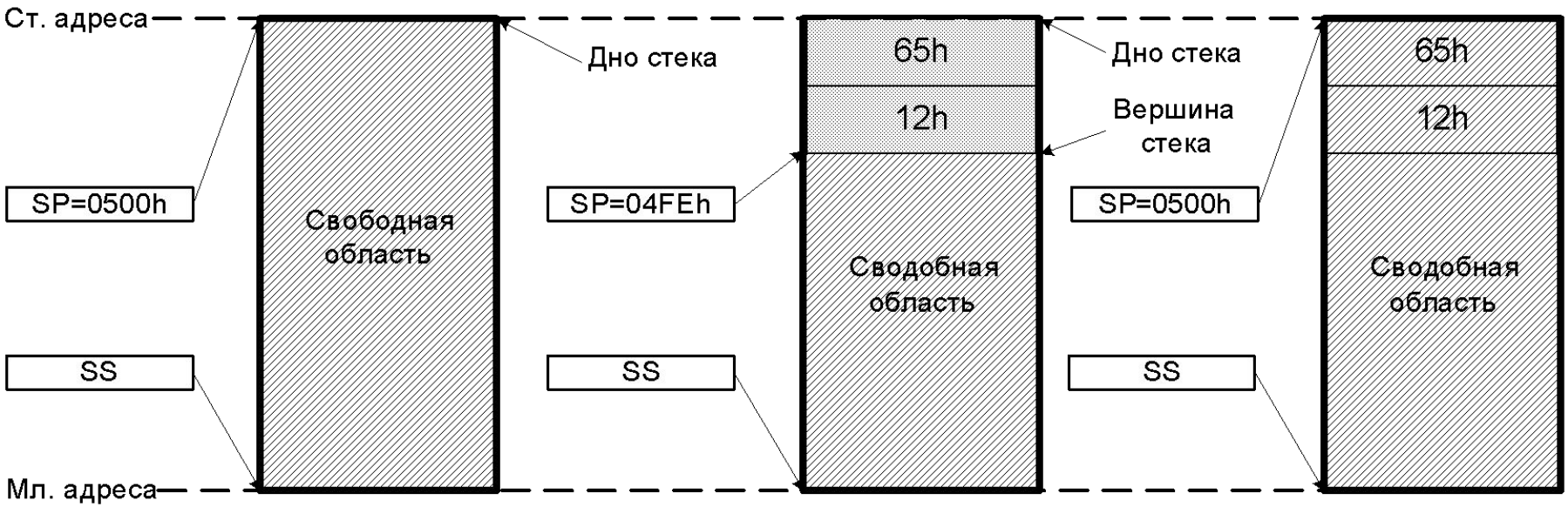
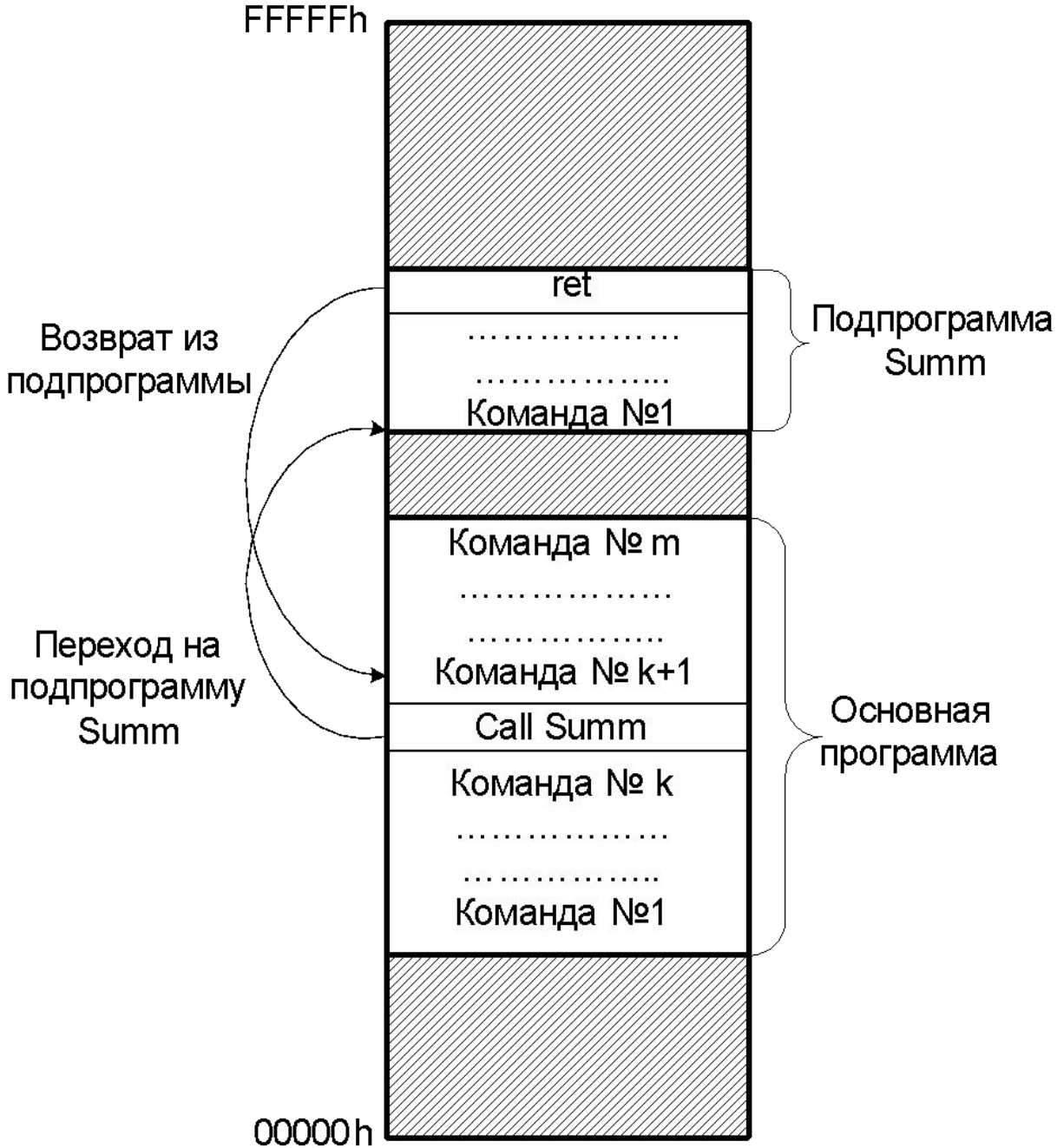
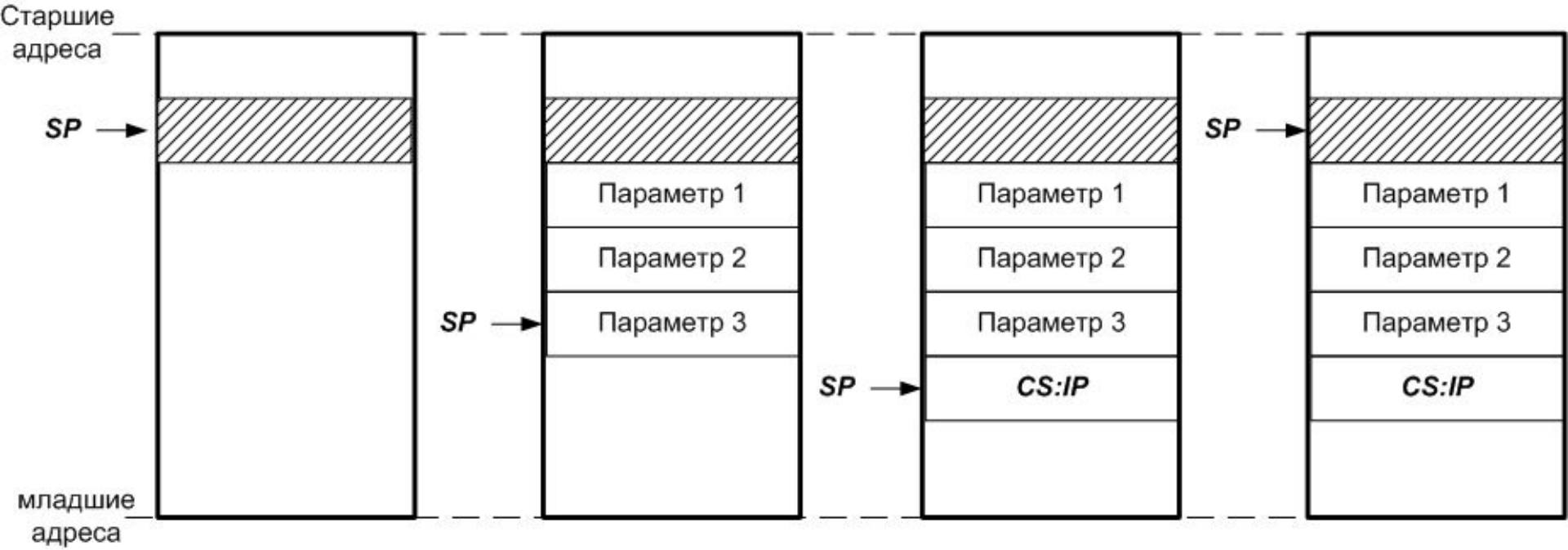


Схема работы со стеком




FFFFFh
ret
…….
……...
Команда №1
Подпрограмма
Summ
Возврат из
подпрограммы
Команда № m
…….
……..
Команда № k+1
Переход на
подпрограмму
Summ
Call Summ
Команда № k
…….
……..
Команда №1
Основная
программа
00000h


Старшие адреса
SP
Параметр 1
Параметр 2
Параметр 3
SP
Параметр 1
Параметр 2
Параметр 3
CS:IP
SP
Параметр 1
Параметр 2
Параметр 3
CS:IP
SP
младшие адреса

```
; add your code here
push 150            ; передача параметра 1
push 150            ; передача параметра 2
push 30             ; передача параметра 3
call myFunction   ; вызов подпрограммы MyFunction
HLT         ; halt!  ; остановка работы процессора
myFunction proc ;подпрограмма MyFunction
    push bp         ; сохраняем предыдущее значение
    mov bp,sp               ; запоминаем в регистре BP адрес последнего элемента стека
    xor ah,ah               ;обнуляем старший байт регистра AX (AH)
    mov al,[bp+4]  ; записываем в регистр AL значение  параметра 3
    add al,[bp+6]   ; складываем содержимое регистров AL и параметра2
; и сохраняем результат в AL
    jnc l1              ; если есть переполнение регистра AL,
    inc ah              ; то увеличиваем регистр AH на 1
l1:                     ;метка
    add al,[bp+8]   ; прибавляем к регистру AL содержимое параметра 1
    jnc l2:             ; если есть переполнение регистра AL,
    inc ah              ; то увеличиваем регистр AH на 1
                ; в итоге в регистре AX будет содержать сумму 3 параметров
l2:
    mov bl,3        ; помещаем в регистр BL значение 3
    div bl              ; делим содержимое регистра AX на BL
                ; в регистре AL - будет целая часть результата операции
                ;деления, в AH -остаток.
    pop bp          ; восстанавливаем содержимое регистра BP
    ret 6               ; вытаскиваем из стека адрес возврата и устанавливаем
;SP=SP+6
myFunction endp
```