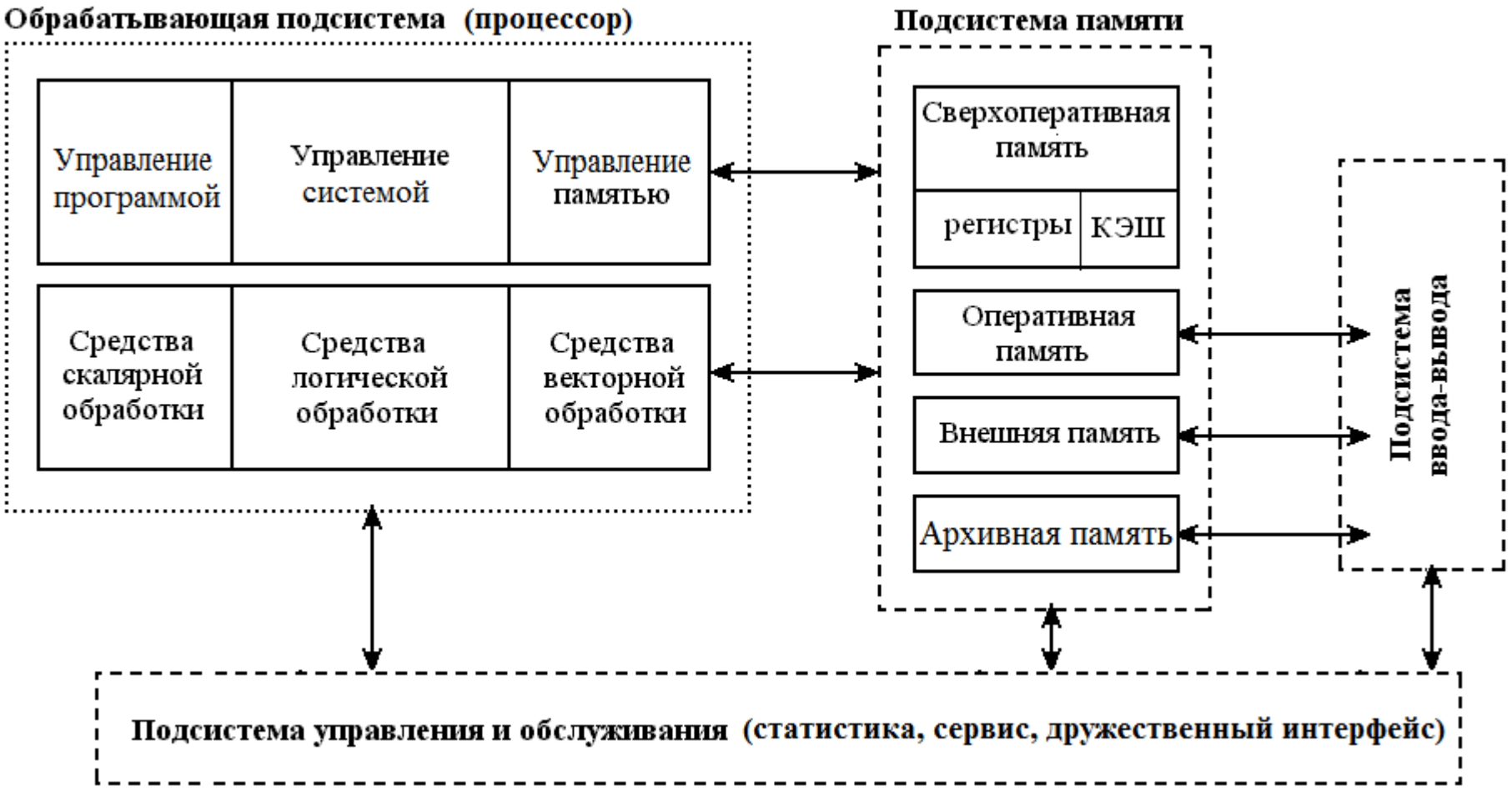
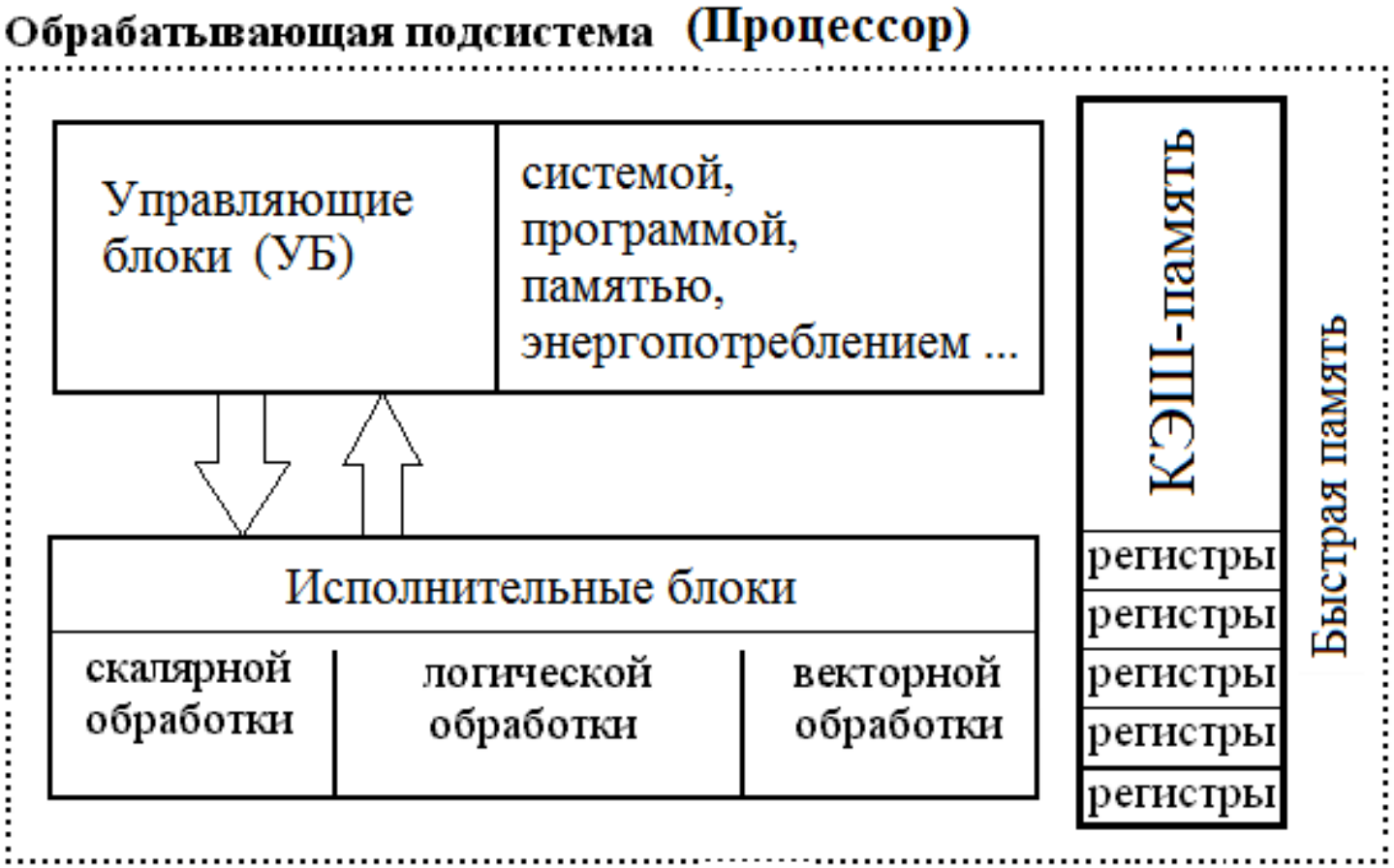
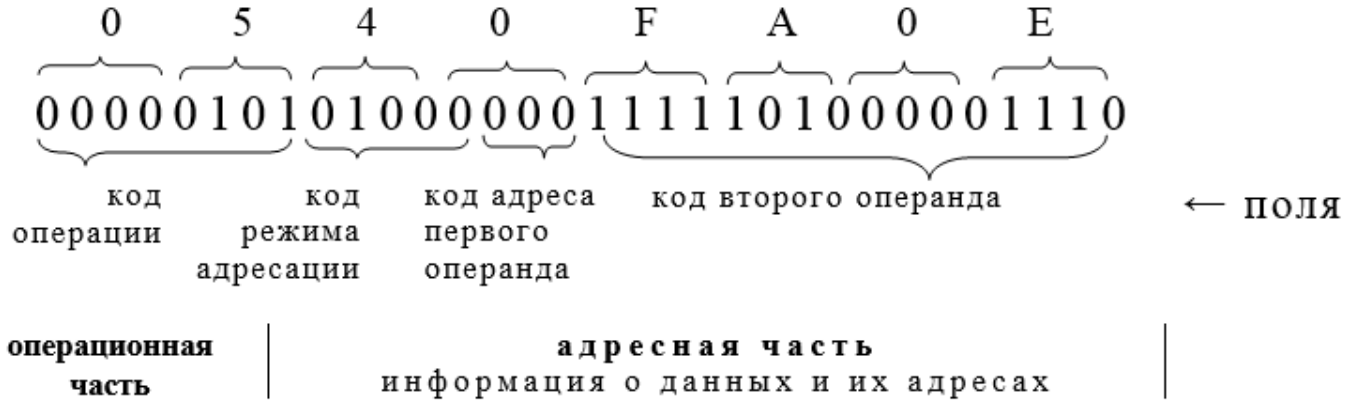
Функциональная схема ВС



УПРОЩЕННАЯ ФУНКЦИОНАЛЬНАЯ СХЕМА ПРОЦЕССОРА

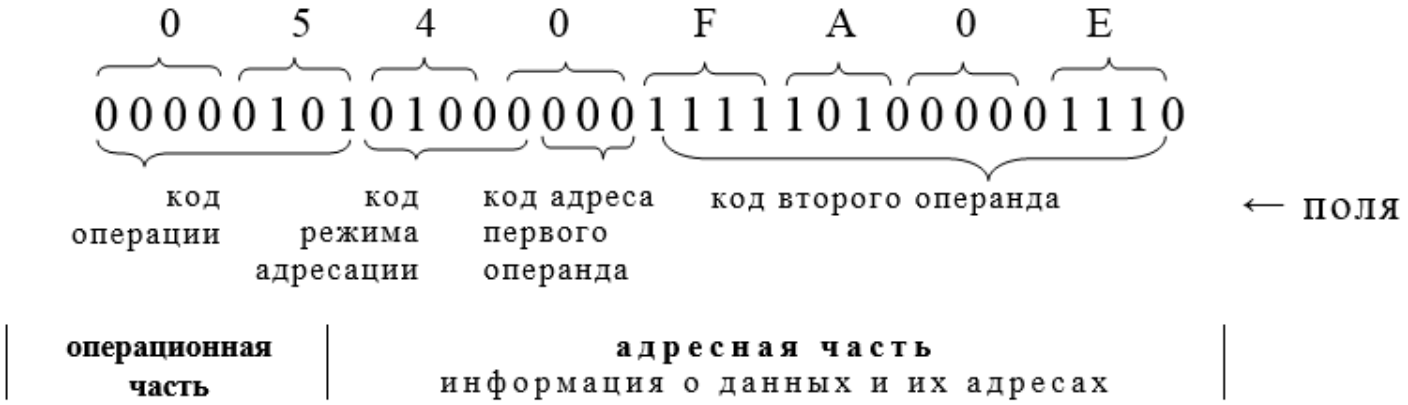


ПОРЯДОК ВЫПОЛНЕНИЯ ОДНОЙ МАШИННОЙ ИНСТРУКЦИИ



- Формирование адреса инструкции,
- Чтение инструкции из памяти
- Расшифровка кода инструкции и разбиение инструкции на мопы
- Формирование адреса данных,
- Чтение операндов (данных из памяти/регистров)
- Выполнение над операндами действий, закодированных мопами
- Сохранение результата операции (и признаков) по адресу, определяемому в инструкции
- Проверка исключений и прерываний появившихся во время исполнения инструкции

ПОРЯДОК ВЫПОЛНЕНИЯ ОДНОЙ МАШИННОЙ ИНСТРУКЦИИ



- Формирование адреса инструкции,
- Чтение инструкции из памяти
- Расшифровка кода инструкции (разбиение на мопы)
- Формирование адреса данных,
- Чтение операндов (данных из памяти/регистров)
- Выполнение над операндами действий, закодированных мопами
- Сохранение результата операции (и признаков) по адресу, определяемому в инструкции
- Проверка исключений и прерываний появившихся во время исполнения инструкции

ПОРЯДОК ВЫПОЛНЕНИЯ ОДНОЙ МАШИННОЙ ИНСТРУКЦИИ

Фаза 1. Выборка инструкции – формирование адреса инструкции, чтение инструкции из памяти

Фаза 2. Декодирование – расшифровка кода инструкции и разбиение инструкции на мопы (или формирование управляющих сигналов)

Фаза 3. Выборка операндов – формирование адреса данных, чтение данных из памяти

Фаза 4. Исполнения – выполнение над операндами в исполнительных устройствах процессора действий, закодированных мопами/УС

Фаза 5. Запись результата – сохранение результата операции по адресу, определяемому в инструкции

Фаза 6. Проверка исключений и прерываний появившихся во время исполнения команды

УПРОЩЕННАЯ СТРУКТУРНАЯ СХЕМА ОДНОГО ЯДРА ПРОЦЕССОРА

Фаза 1. *Выборка инструкции*

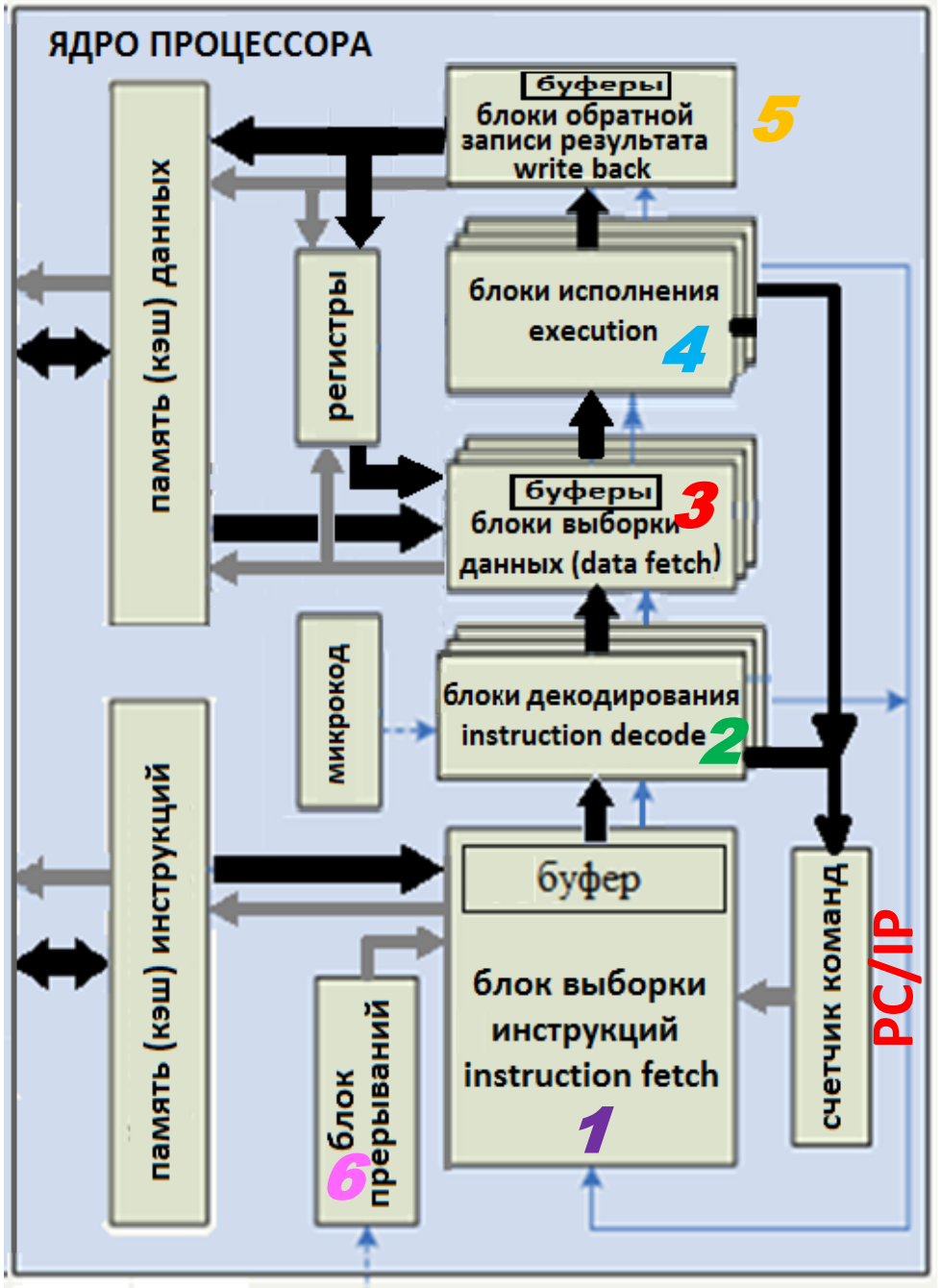
Фаза 2. *Декодирование*

Фаза 3. *Выборка операндов*

Фаза 4. *Исполнения*

Фаза 5. *Запись результата*

Фаза 6. *Проверка исключений и прерываний*



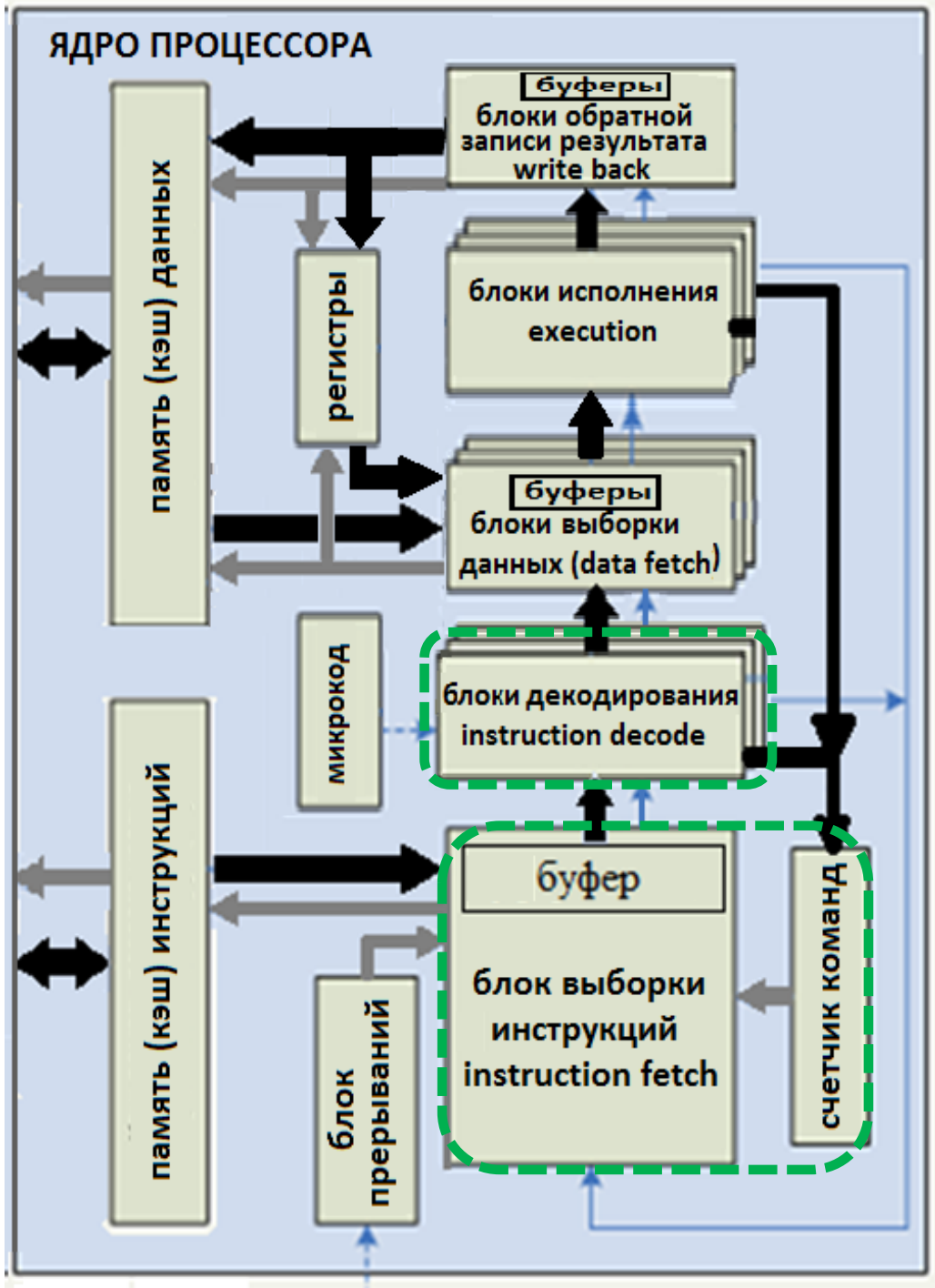
данные

адреса

сигналы управления

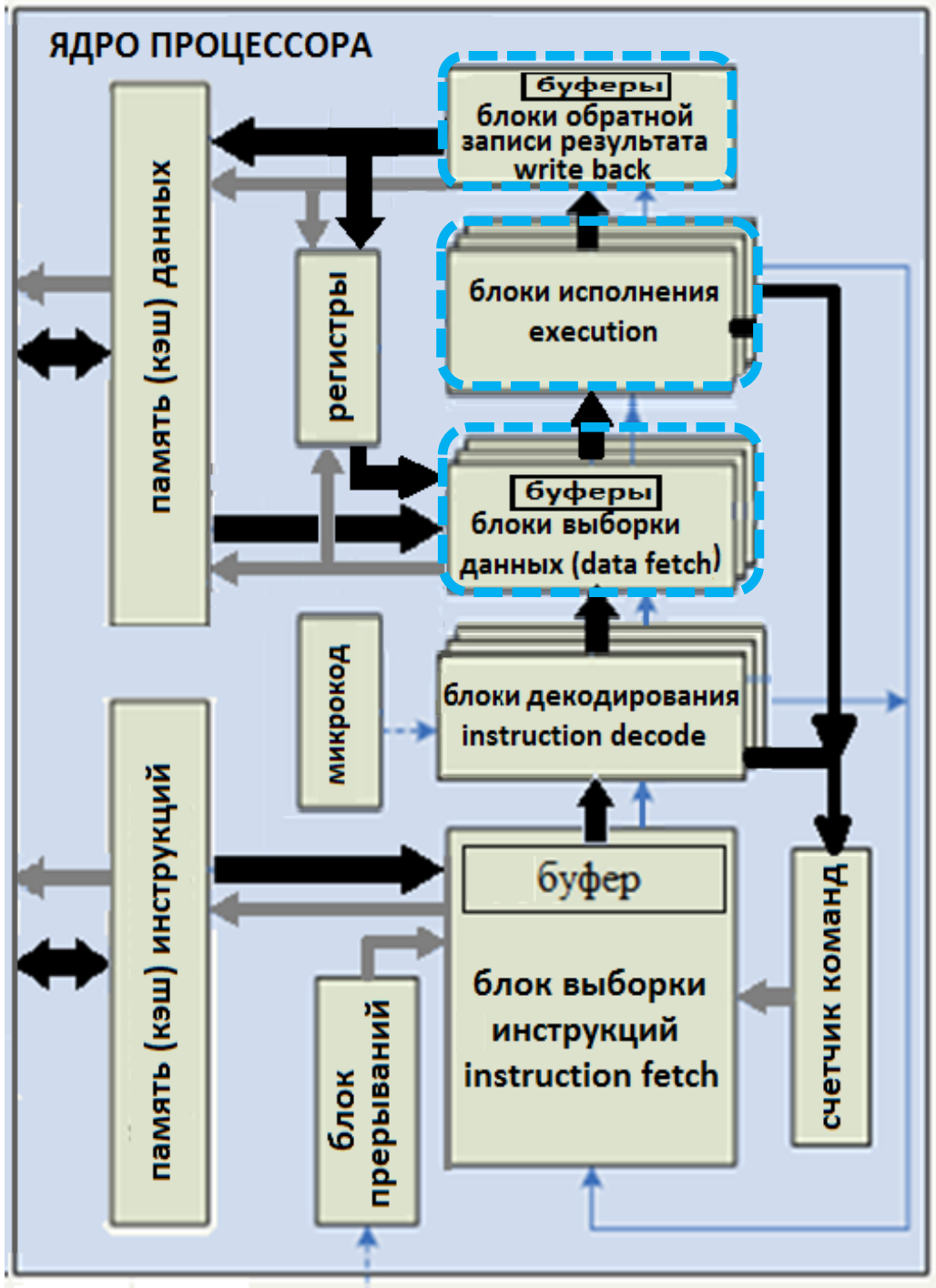
Функции блоков управления программой

- определения адреса следующей инструкции
- считывание инструкции из памяти
- дешифрация инструкции в мопы или управляющие сигналы



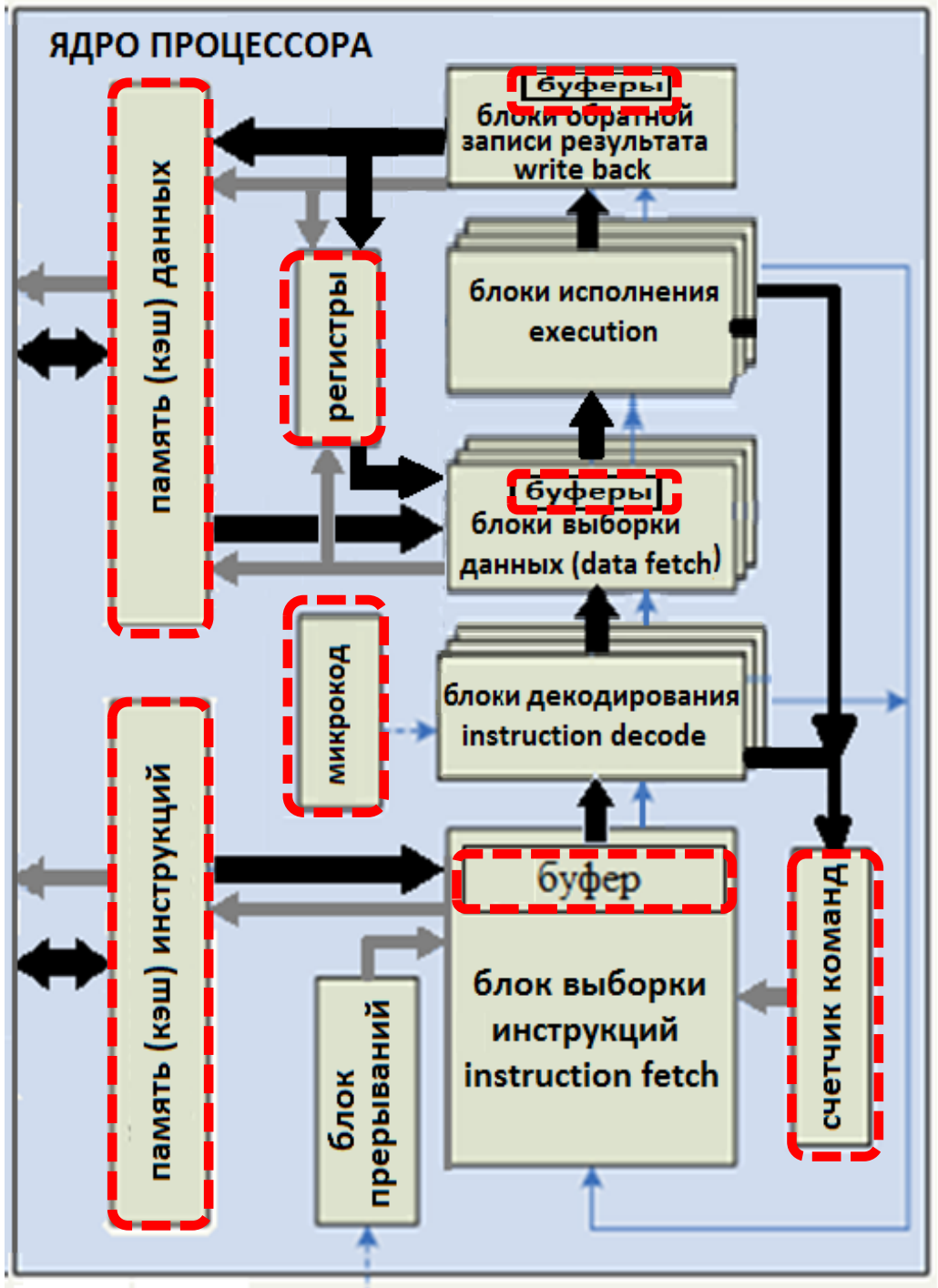
Функции блоков обработки

- выборка данных по инструкциям/мопам
- исполнение операций
- изменение состояния задачи/процессора по результату
- сохранение результата

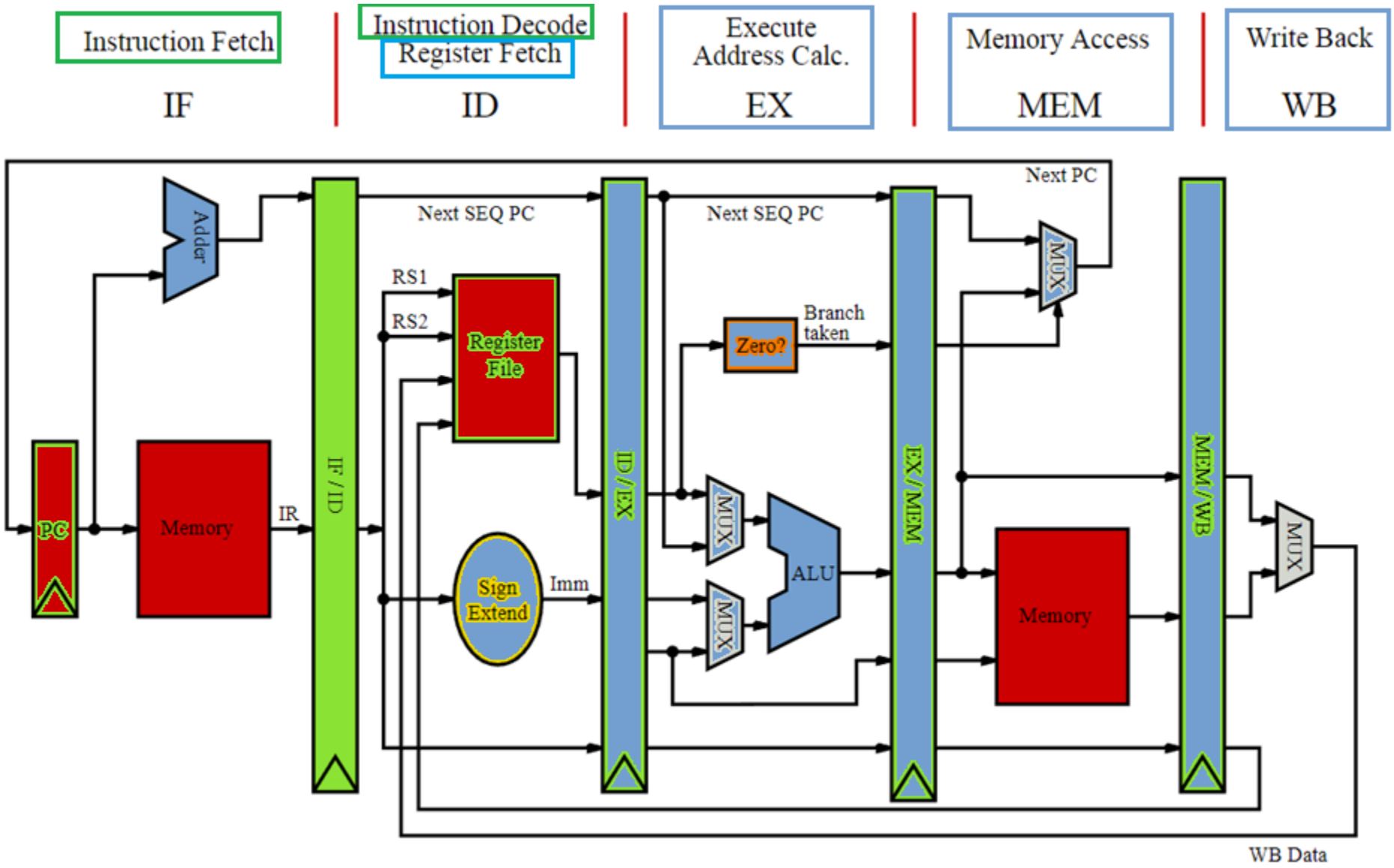


Функции блоков внутренней памяти процессора

временное хранение
- адресов,
- битов состояния (флагов),
- кодов инструкции,
- кодов микроопераций,
- данных,
- результатов

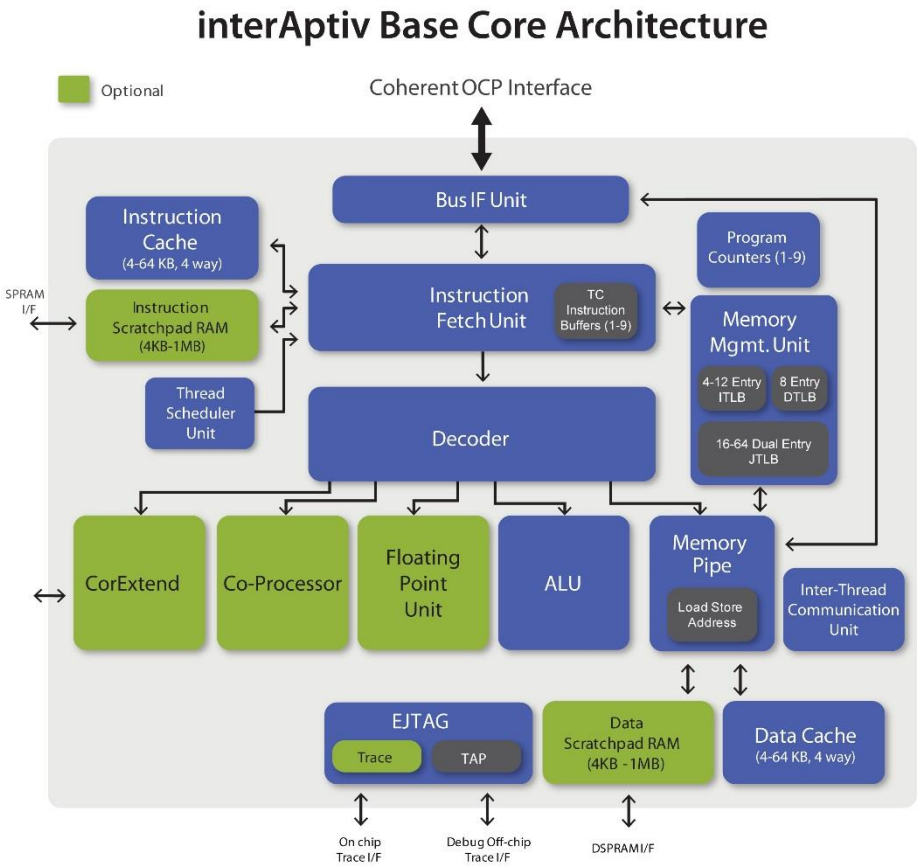


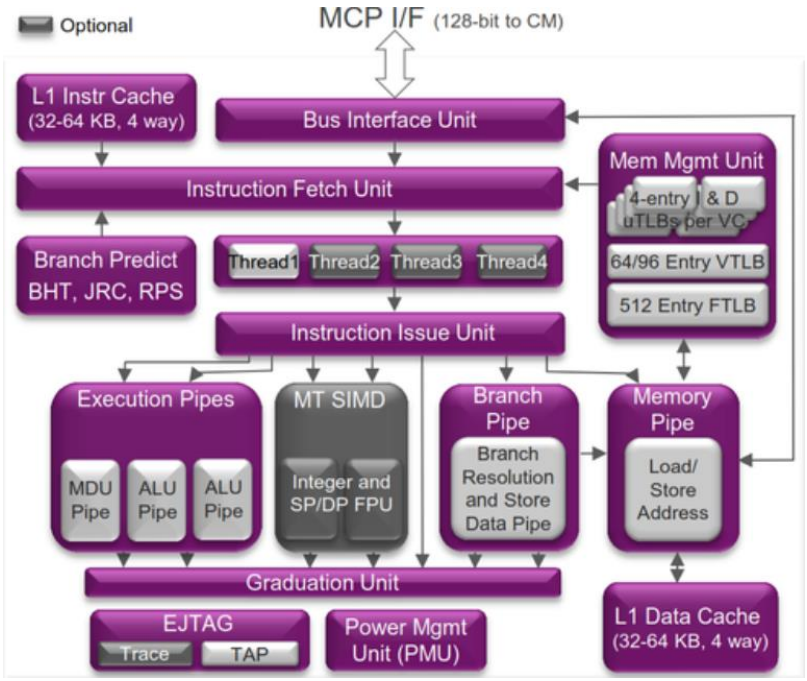
Микроархитектура RISC-процессора MIPS



https://ru.wikipedia.org/wiki/MIPS_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)

Микроархитектура RISC-процессора MIPS





Простая RISC-инструкция = микрооперация

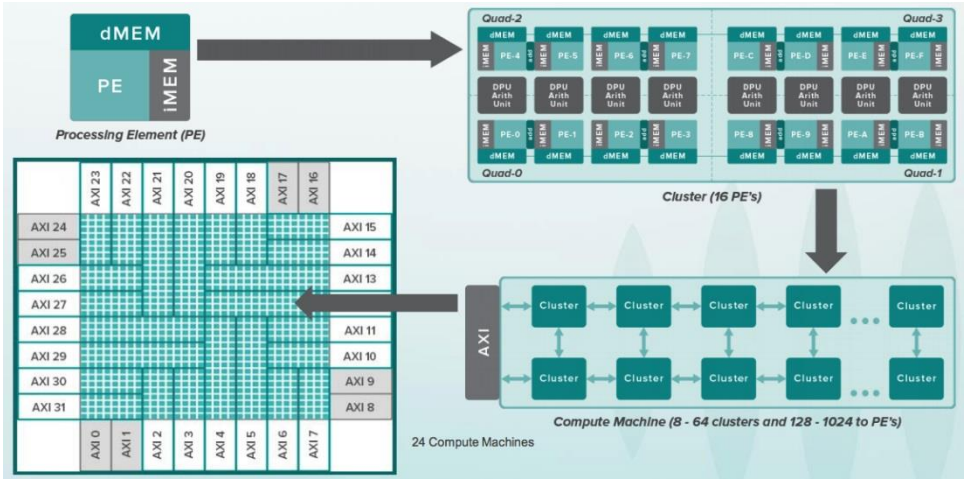
https://ru.wikipedia.org/wiki/MIPS_(%D0%B0%D1%80%D1%85%D0%B8%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0)

Область применения процессора MIPS

несколько миллиардов выпущенных чипов на основе MIPS

- автомобильная электроника DENSO (Тойота)
- аппаратные блоки видео-обработки и алгоритмов распознавания
- Контроллеры в игровых приставках (Nintendo, Sony Playstation)
- рабочие станции Silicon Graphics
- учебные университетские проекты по модификации реального промышленного процессора MIPS
- российский компьютер на основе процессора КОМДИВ-64 (вариант MIPS + векторные расширения)
- процессор Байкал-T1 на основе MIPS P5600 «Apache» используется в российских контроллерах станков и сетевых устройствах
- Wave Computing совместно с Broadcom - сетка (mesh) из неск. десятков(тысяч) процессорных элементов для ускорителя нейросетей (7 нм, выше 6 ГГц)

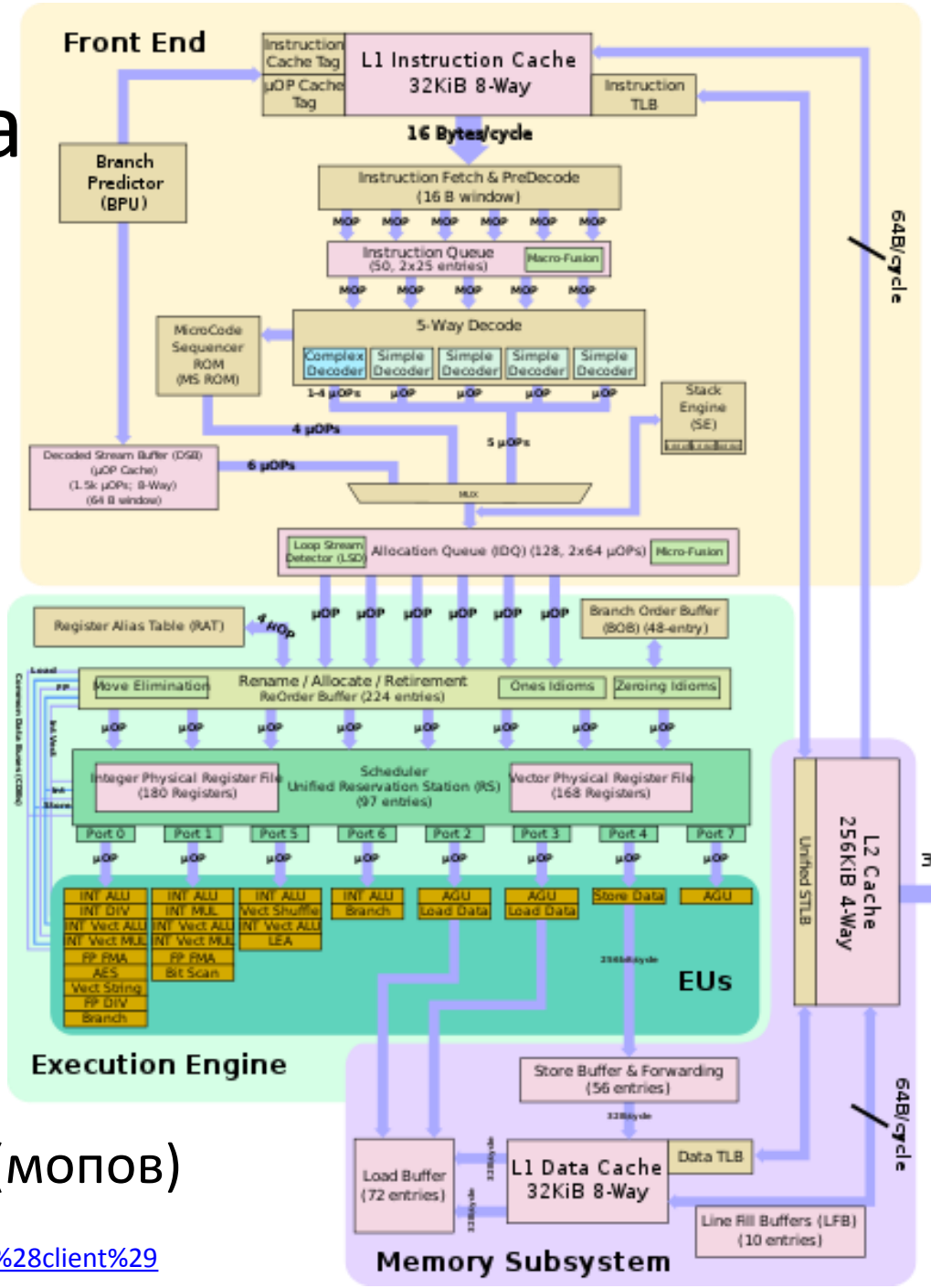
MIPS - Open Source архитектура (с открытым кодом)



https://habr.com/ru/post/433524/

Микроархитектура процессора Intel

- SkyLake
- KabyLake
- AmberLake
- CannonLake
- Comet Lake



Сложная CISC-инструкция = Σ(мопов)

https://en.wikichip.org/wiki/intel/microarchitectures/skylake_%28client%29

Область применения процессора Intel

Лидер индустрии - 80% мирового рынка процессоров (собственный «дизайн» - проектирование + собственное производство)

- ПК: настольные, переносные, планшеты, сматрфоны, ноубуки, нетбуки,
- Рабочие станции
- Серверные платформы
- Встраиваемые решения (основа интернета вещей, интеллектуальные устройства, системы коммуникаций, системы хранения)
- Основа Суперкомпьютеров

Архитектура/программная модель RISC-процессора MIPS64

1 – РОН или GPU (регистры общего назначения) для целых чисел (32 шт по 32/64 бит),

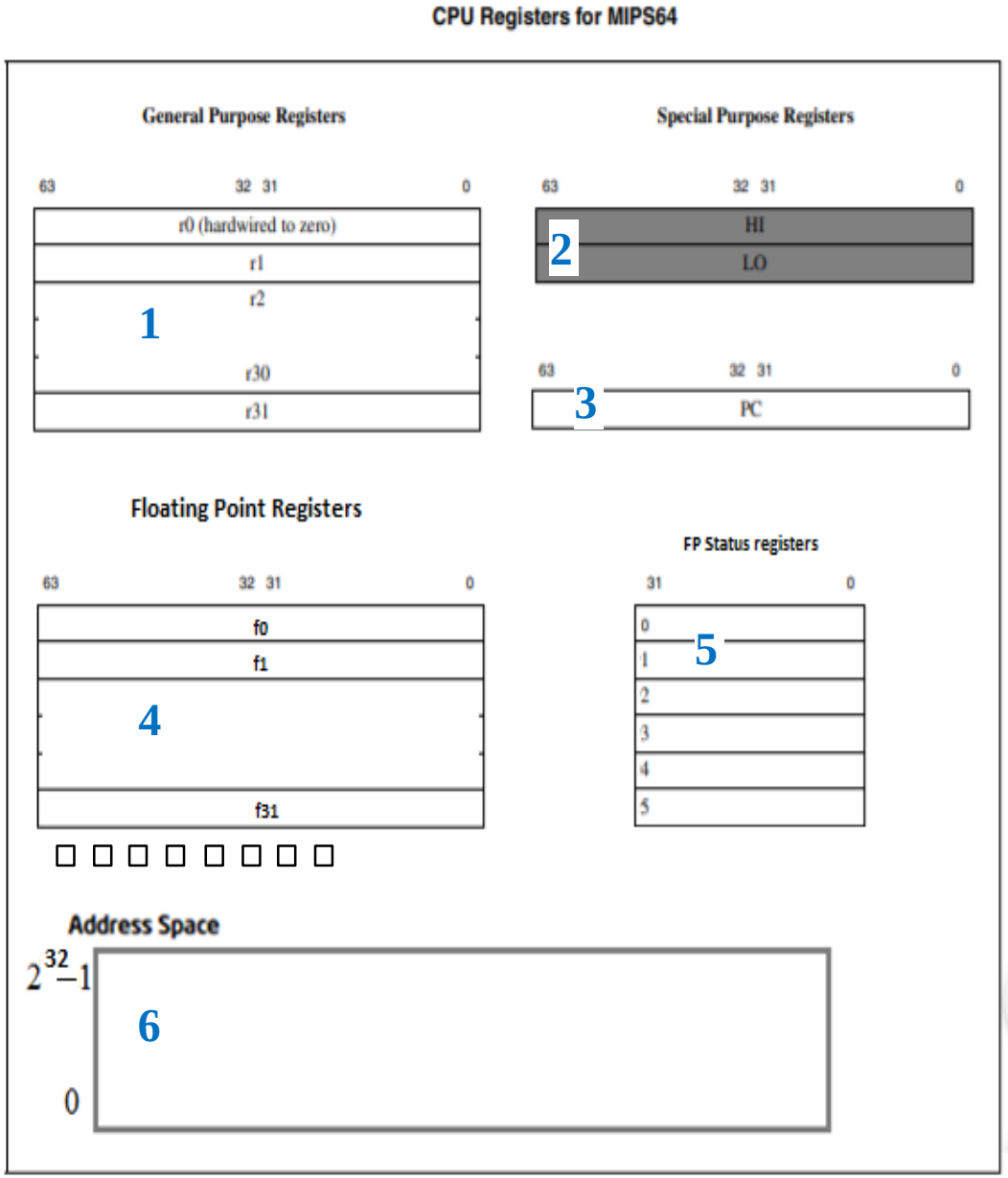
2 – Специальные регистры (hi/lo)

3 – указатель адреса следующей команды (PC)

4 – регистры FPU (32 шт по 32 бит, м.б. использованы парами и превратиться в 16 шт по 64 бит) + флаги условий

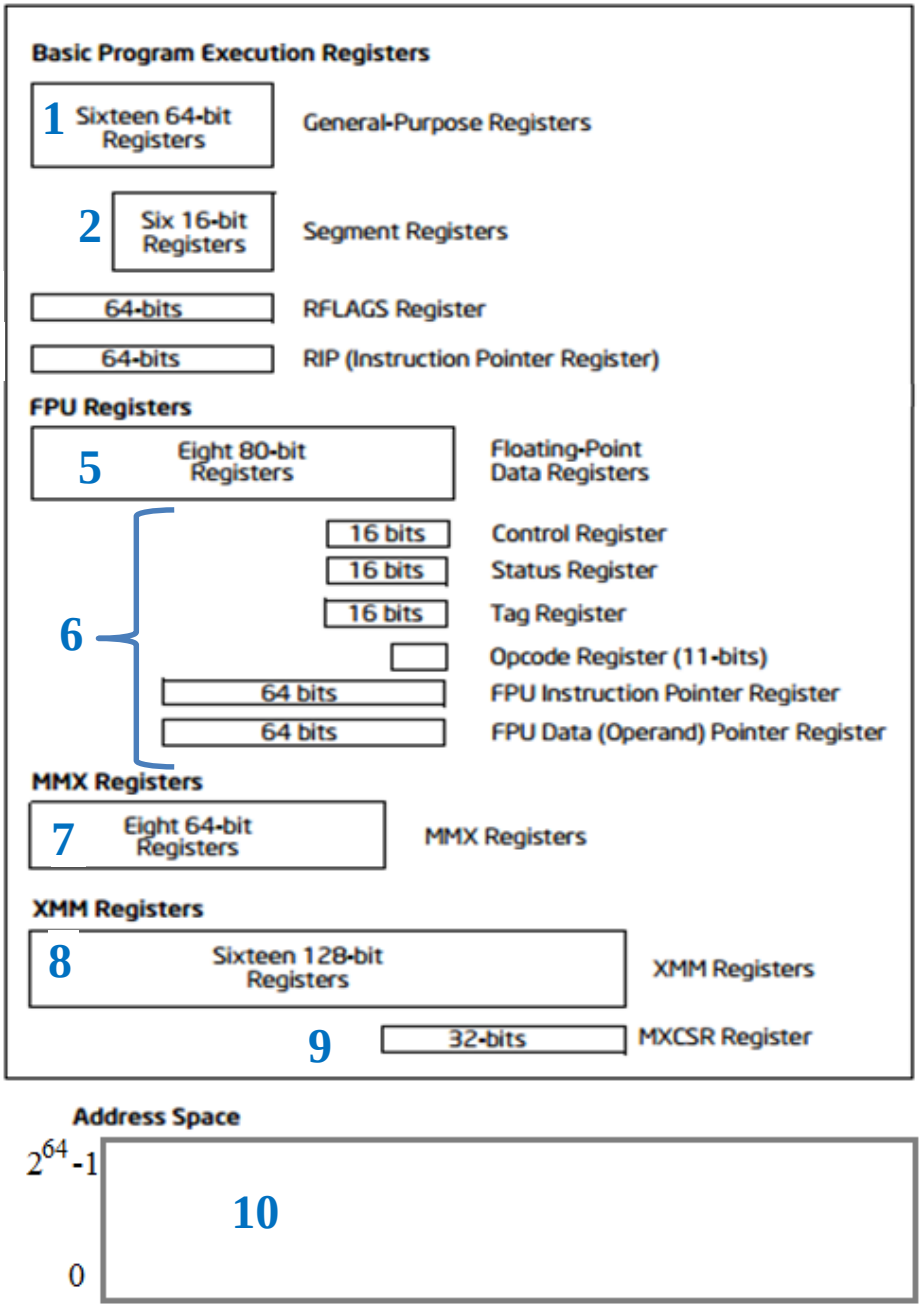
5 – управляющие регистры блока FPU (выбор режима округления, контроля денормализованных параметров, проверки исключений,…)

6 – Оперативная память



https://s3-eu-west-1.amazonaws.com/downloads-mips/documents/MD00083-2B-MIPS64INT-AFP-06.01.pdf

Архитектурная или программная модель Intel 64



1 – РОН или GPU (регистры общего назначения) для целых чисел (16 шт для I64 и 8 шт для I32),

2 – Сегментные регистры

3 – флаговый регистр (или признаков, или состояния)

4 – указатель адреса следующей команды (IP)

5 – регистры FPU (для дробных чисел формата ЧПЗ) 8 шт.

6 – управляющие регистры блока FPU

7 – регистры MMX (векторные целые)

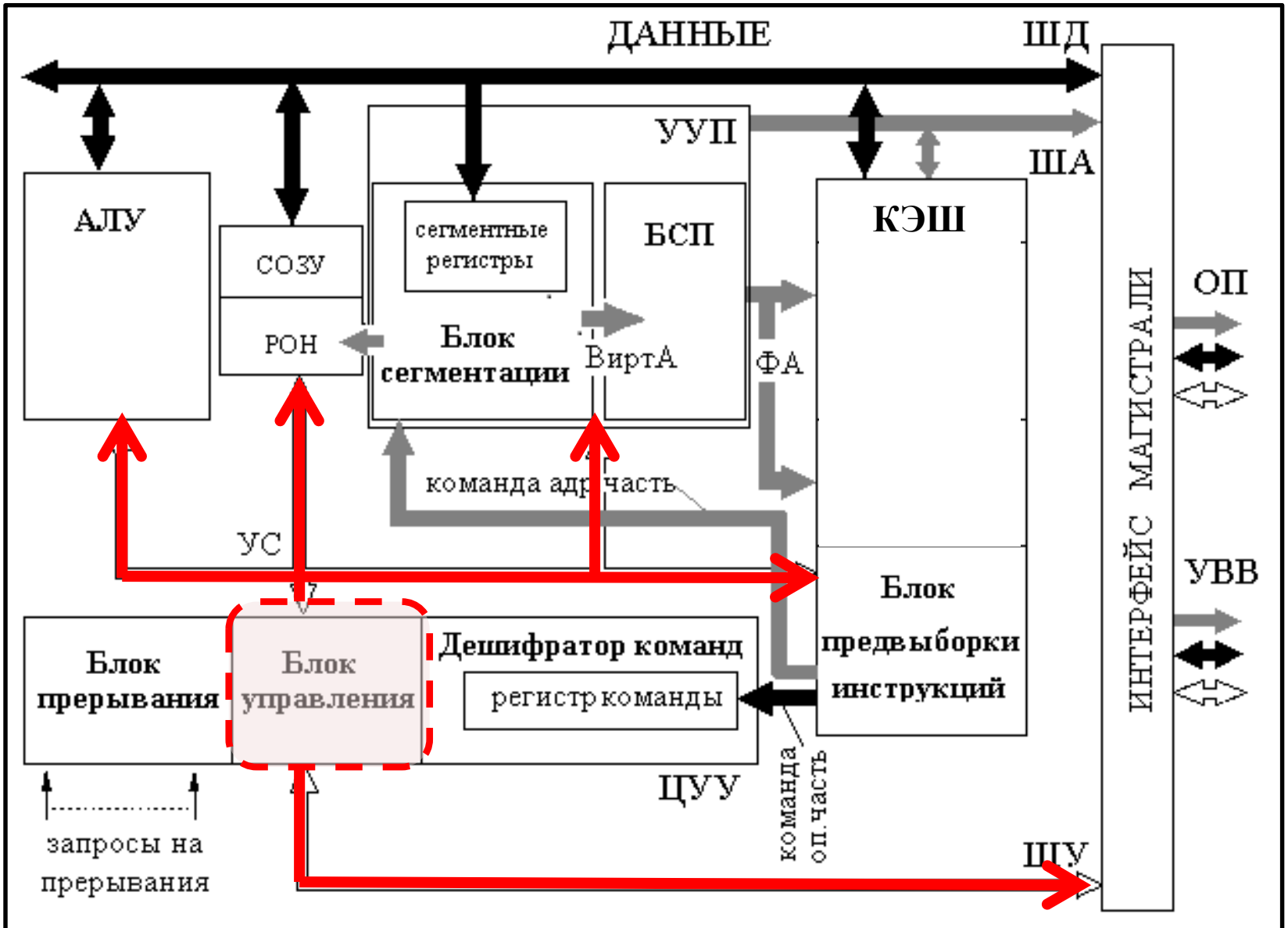
8 – регистры XMM (векторные дробные)

9 – управляющий регистр (статуса и признаков блока SIMD)

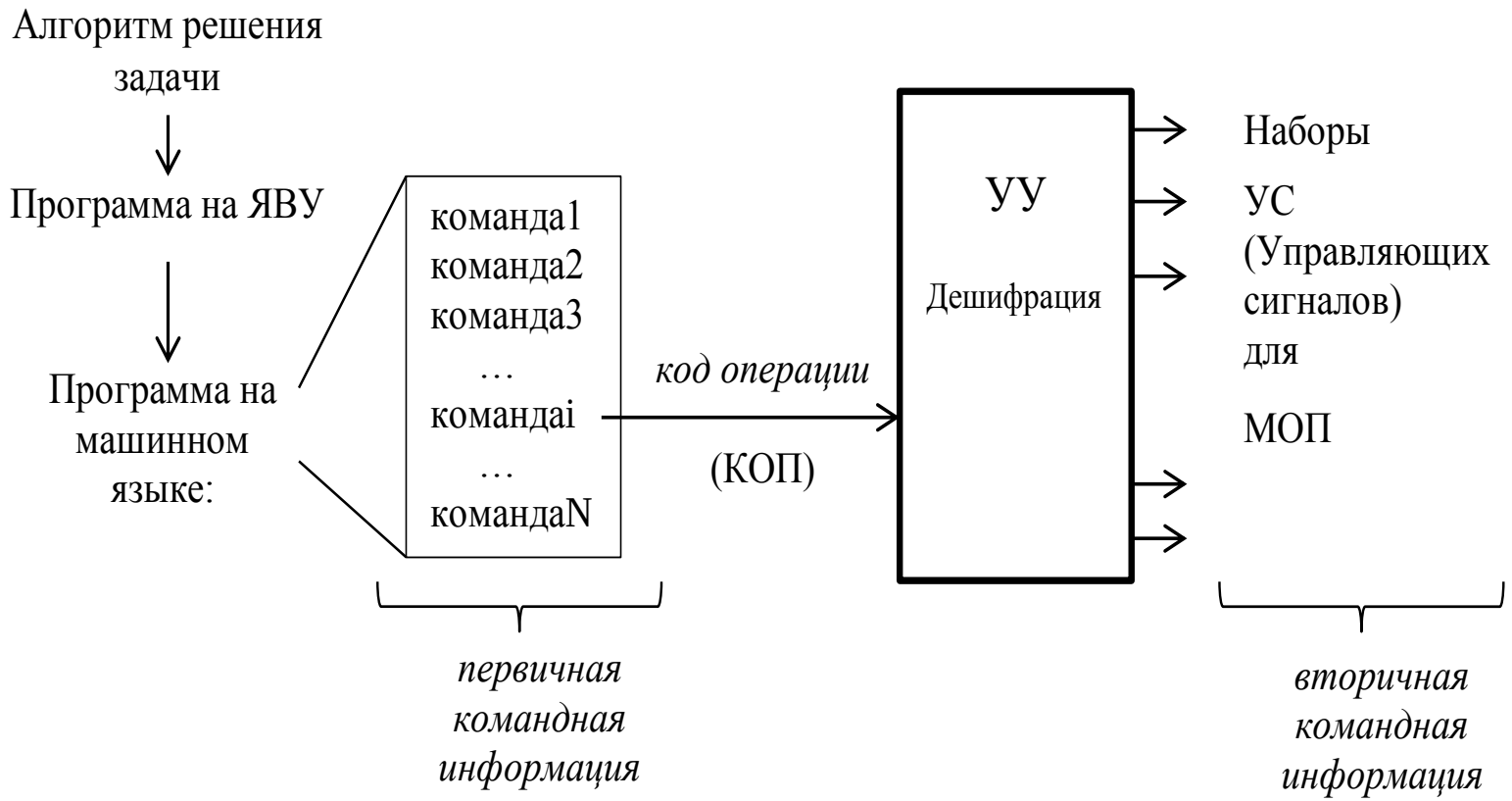
10 – Оперативная память

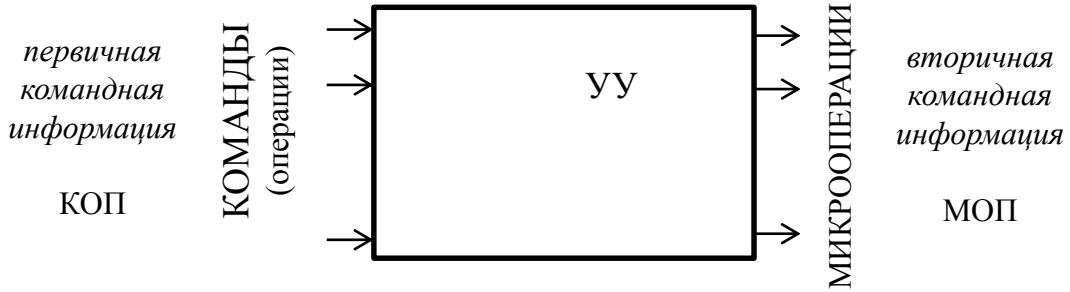
Структурная схема одноядерного процессора

выработка управляющих сигналов

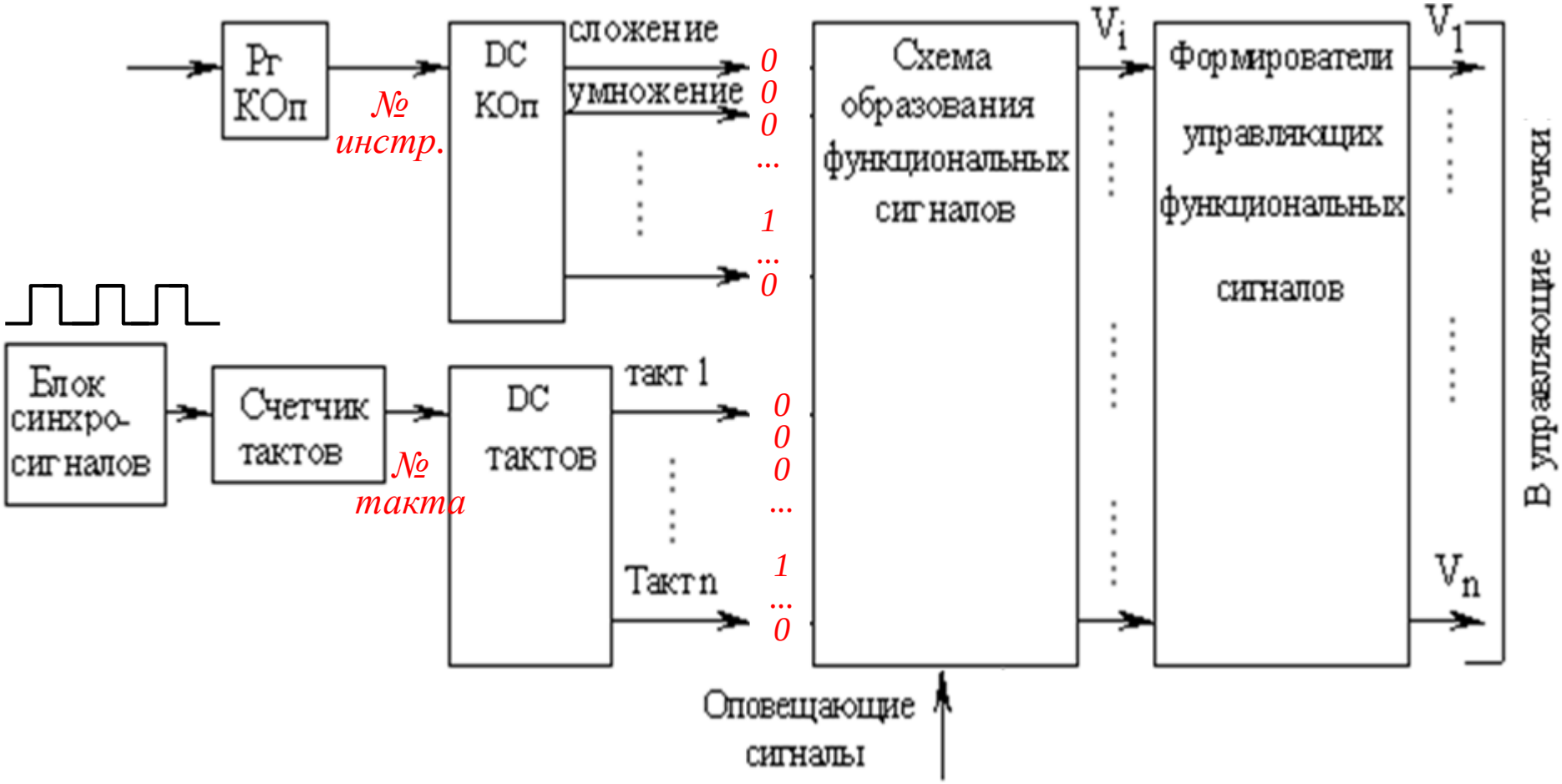


ПРИНЦИП УПРАВЛЕНИЯ РАБОТОЙ ПРОЦЕССОРА





АППАРАТНЫЙ ПРИНЦИП УПРАВЛЕНИЯ ПРОЦЕССОРА (АВТОМАТ С ЖЁСТКОЙ ЛОГИКОЙ)

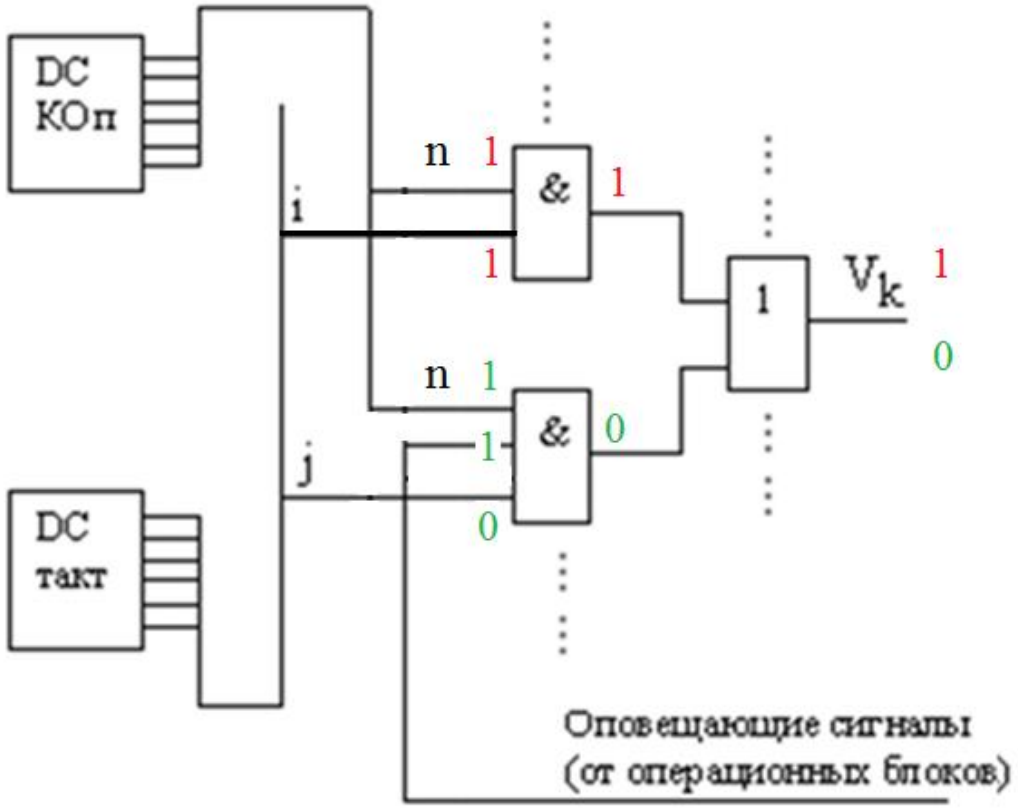


структурная схема автомата

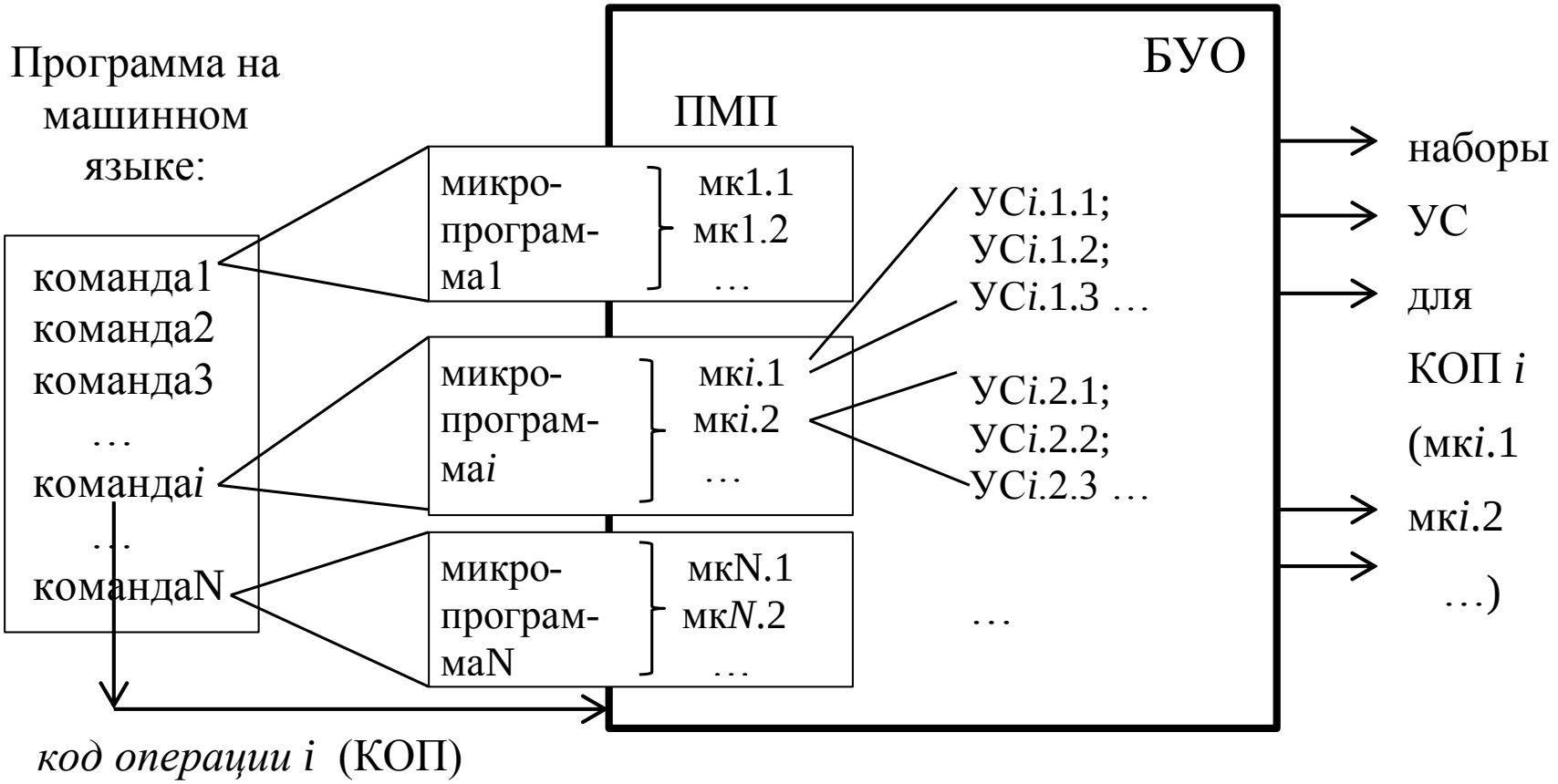
фрагмент схемы, обеспечивающей выработку управляющего сигнала Vk в i-м и j-м тактах выполнения n-й команды

В первом случае
Коп=n, такт=i => на выходе верхнего коньюнктора «1» на выходе нижнего коньюнктора «0», логическое ИЛИ дает «1»
V_k=1 сигнал вырабатывается

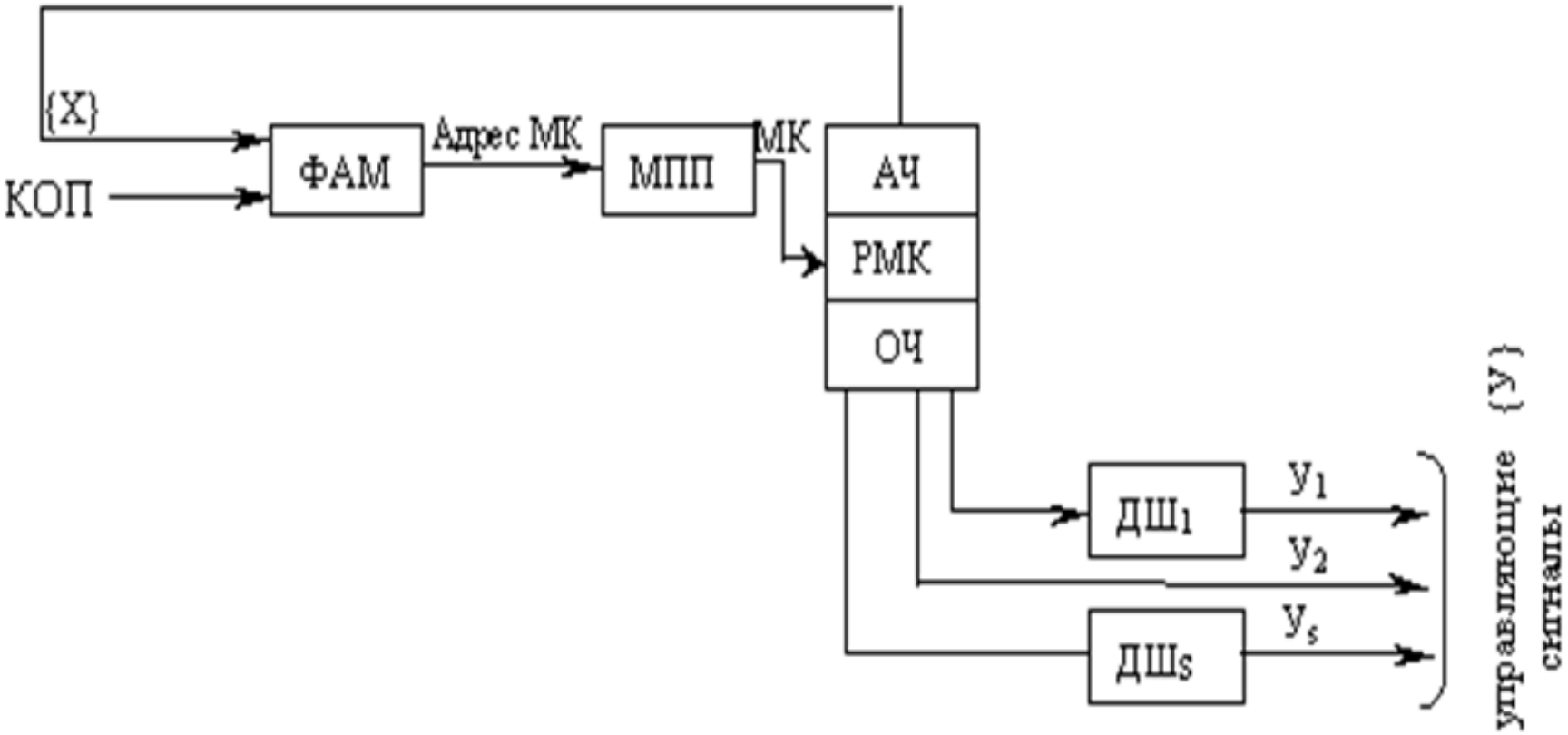
Во втором случае
Коп=n, такт=j и нужный оповещающий сигнал(например флаг)=0, на выходе нижнего коньюнктора «0», на выходе верхнего коньюнктора «0» - не тот такт, логическое ИЛИ дает «0»
Vk=0 сигнал не вырабатывается



УУ ПРОЦЕССОРА С МИКРОПРОГРАММНОЙ ЛОГИКОЙ



Обобщенная структура блока микропрограммного управления (БМУ)



Сравнение УУ процессора с микропрограммной и аппаратной логикой

- Аппаратно=быстро → *высокая скорость (RISC)*
- Программно=медленно → *низкая скорость (CISC)*

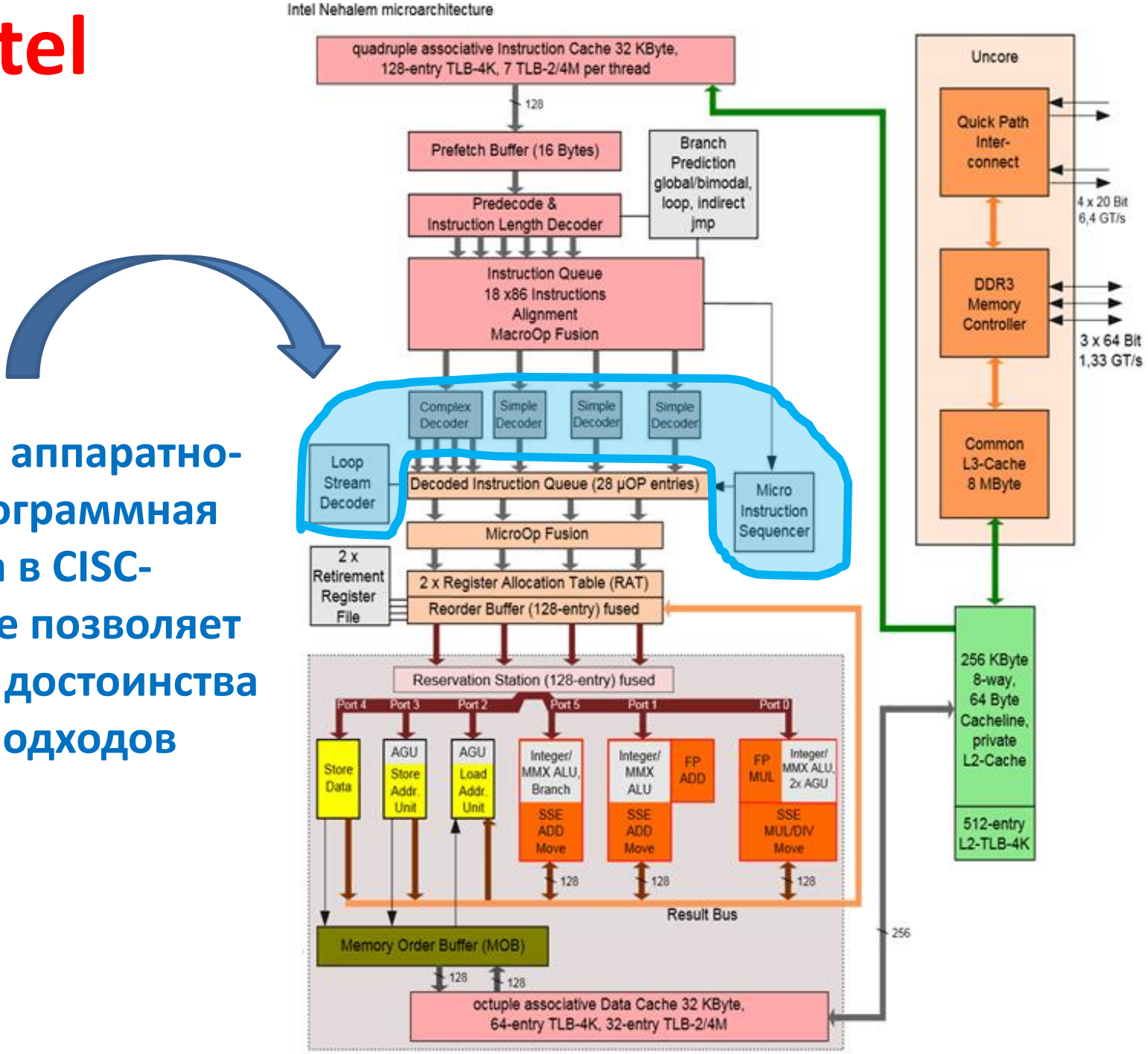
- Создать схему для управления сложными циклическими многотактными командами сложно → *аппаратное управление только для простых команд, а для сложных – микропрограммное управление*

- Изменить логику/добавить новые команды → *полное проектирование нового процессора с аппаратным управлением и минимальное изменение микрокода в памяти микропрограмм для микропрограммного управления*

Совместить достоинства обоих подходов

Intel

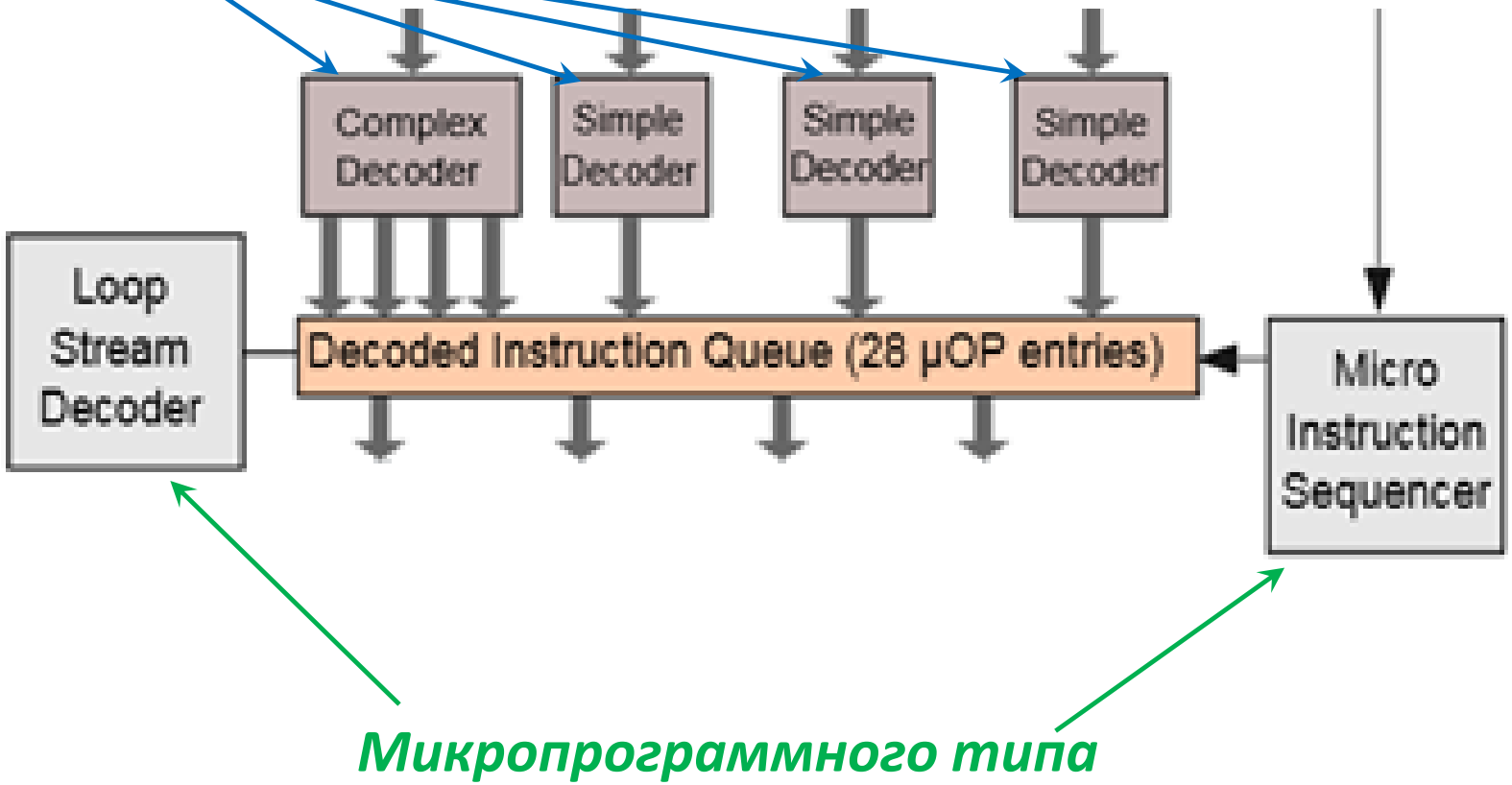
Гибридная аппаратно-микропрограммная логика в CISC-процессоре позволяет совместить достоинства обоих подходов



Совместить достоинства обоих подходов

микропрограммную и аппаратную логику

Аппаратного типа *декодеры=формирователи УС для моп*



Микропрограммного типа

Оценка централизованного управления процессором

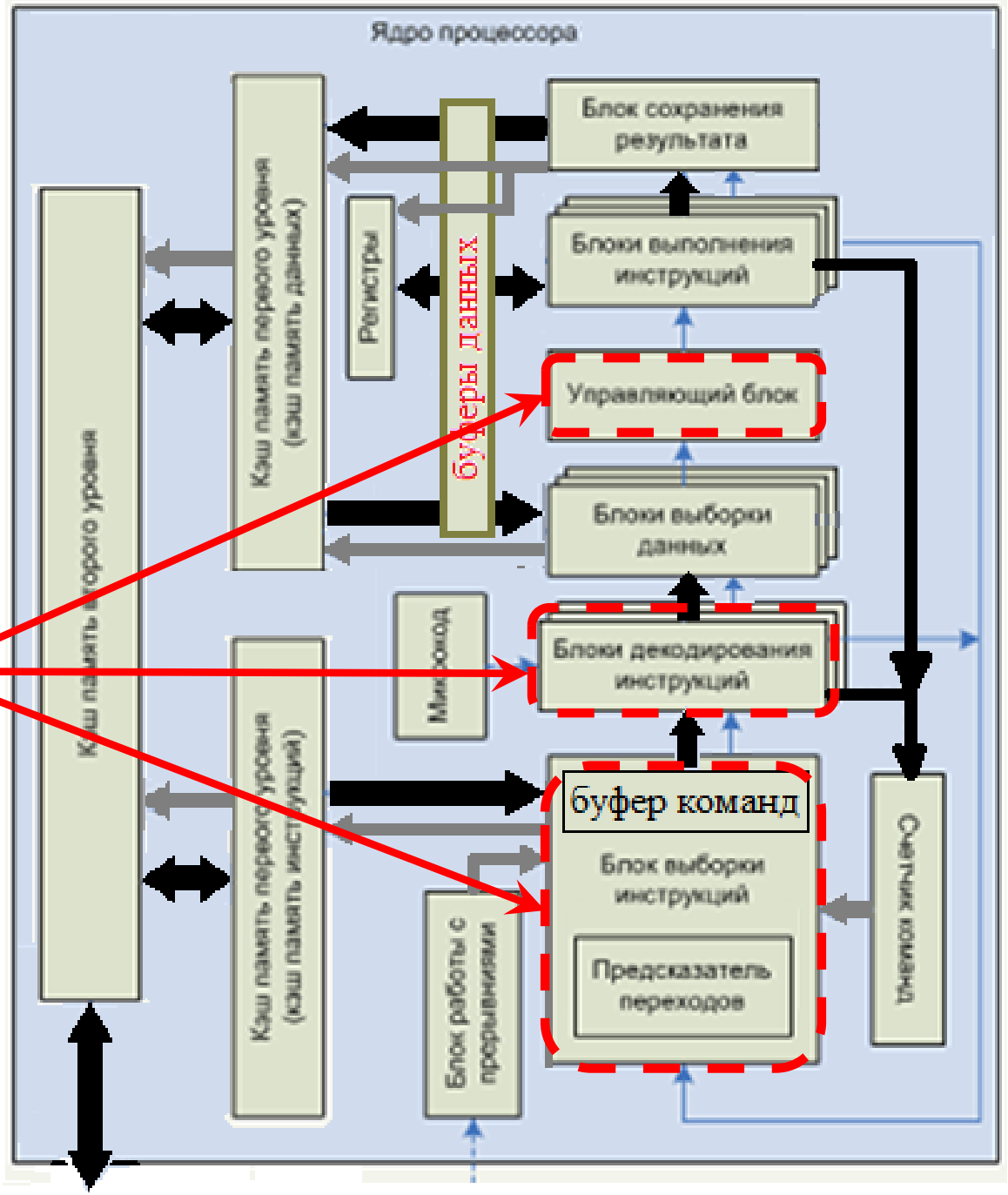
Одно устройство управления – узкое место

снижение скорости

требуется несколько управляющих блоков

Структурная схема одного ядра процессора с *децентрализованным управлением*

ядро =
блоки управления
+ исполнительные блоки +
внутренняя память

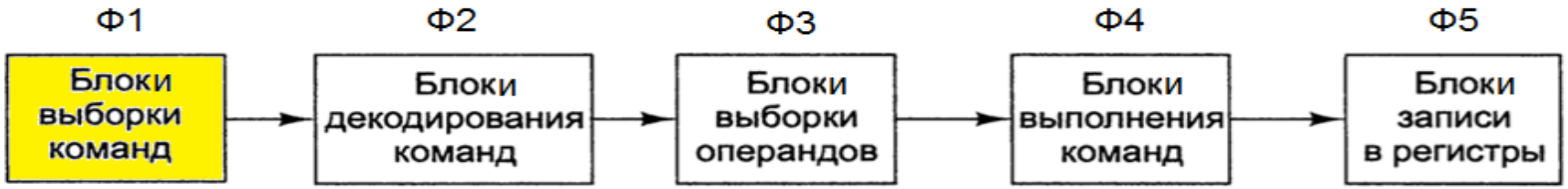


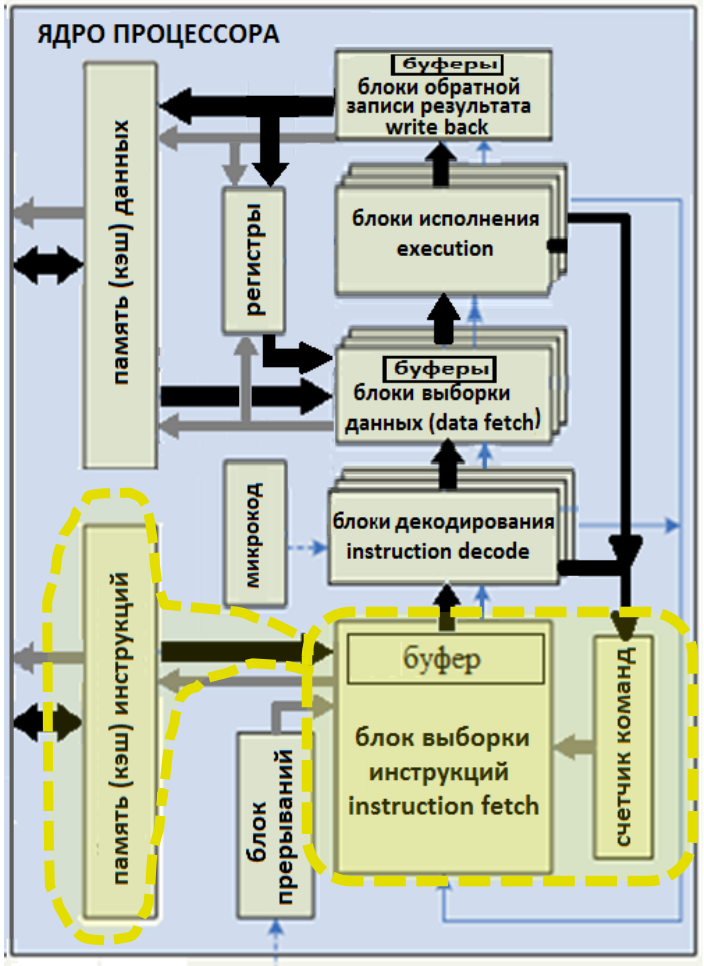
Порядок исполнения потока команд (фазы цикла исполнения команды)

- определения адреса команды — Ф1
- считывание команды
- дешифрация команды в мопы/в УС — Ф2
- выборка операндов (данных по командам/мопам) — Ф3
- исполнение микроопераций — Ф4
- изменение состояния задачи по результату
- сохранение результата — Ф5

Для каждой команды все фазы должны выполняться последовательно

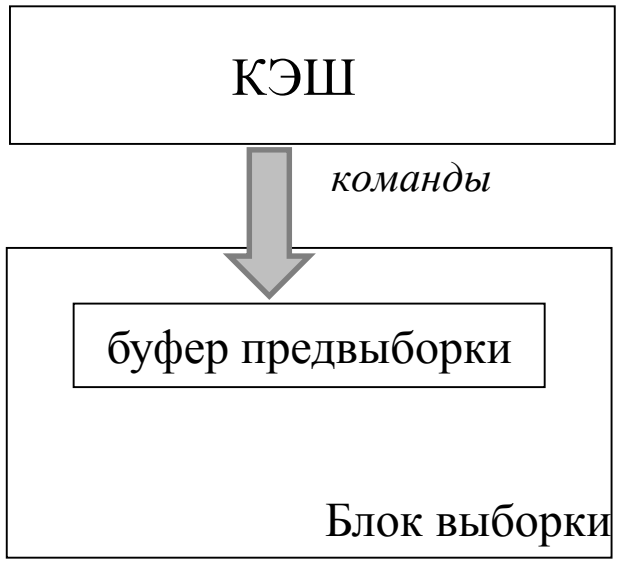
БЛОКИ ВЫБОРКИ КОМАНД (Фаза1)





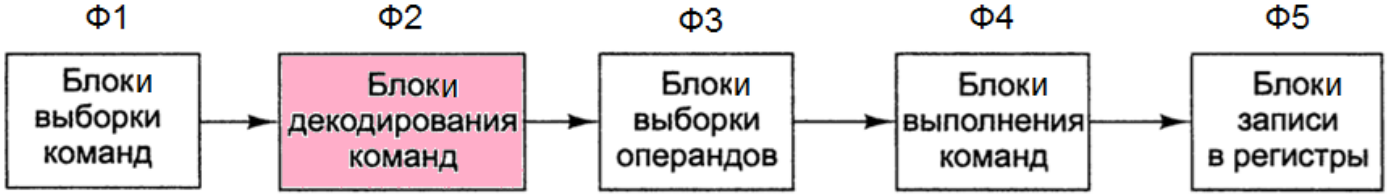
Структурная схема

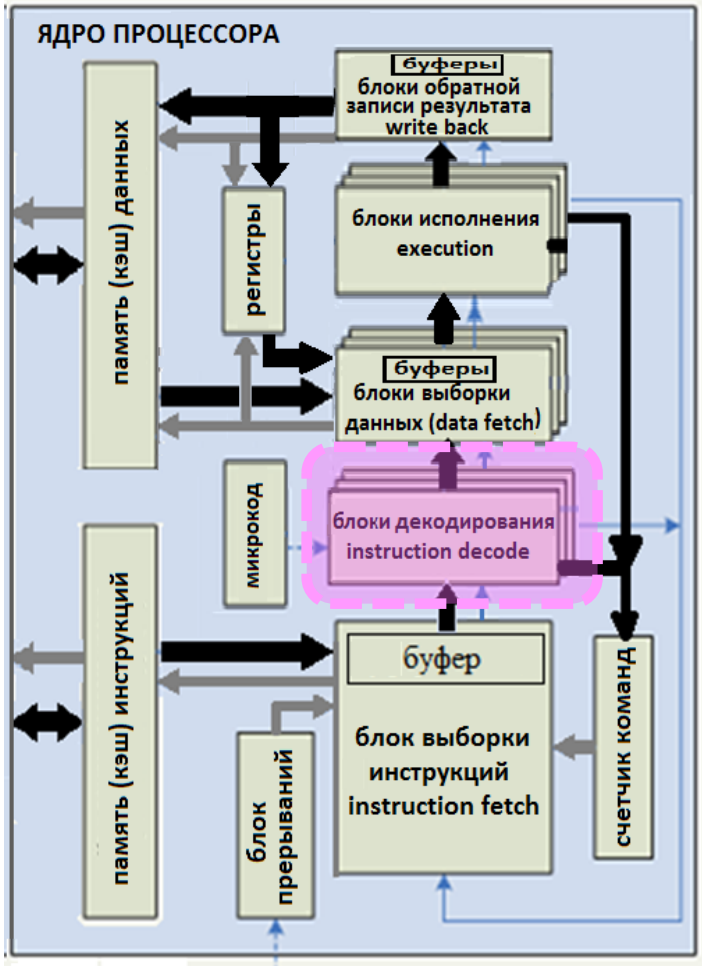
- *Формирование адреса команды (на основании IP)*
- *Поиск блока в КЭШ (долго)*
- *Считывание команды в буфер предвыборки*



Функциональная схема

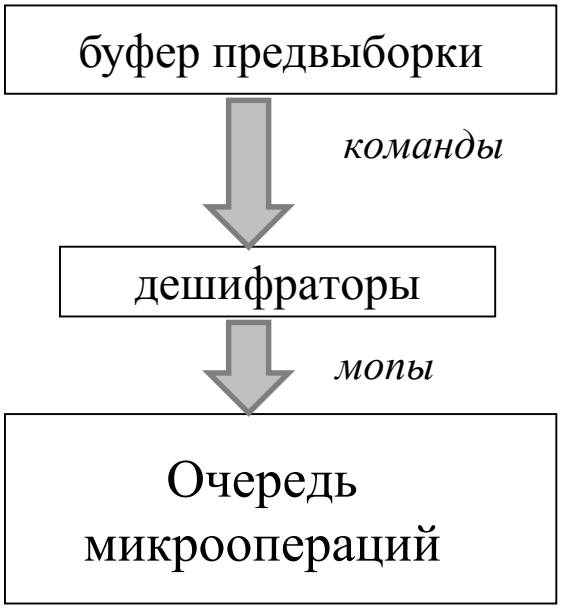
БЛОКИ ВЫБОРКИ КОМАНД (Фаза1)





Структурная схема

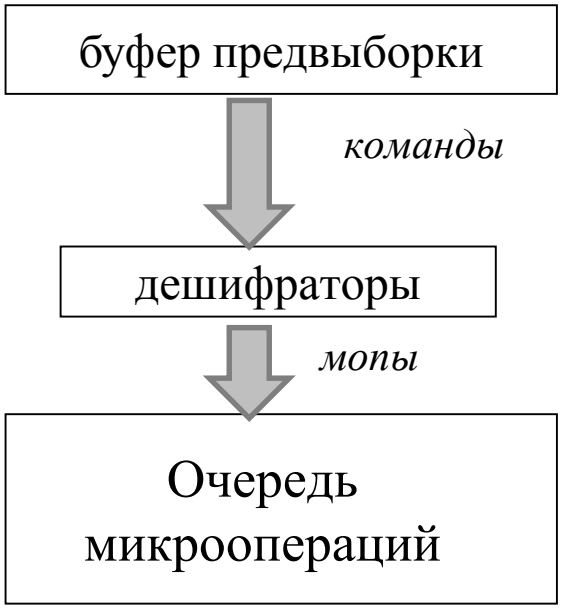
- *Расшифровка кодов операции, префиксов, адресов*
- *Формирование кодов микроопераций (моп)*

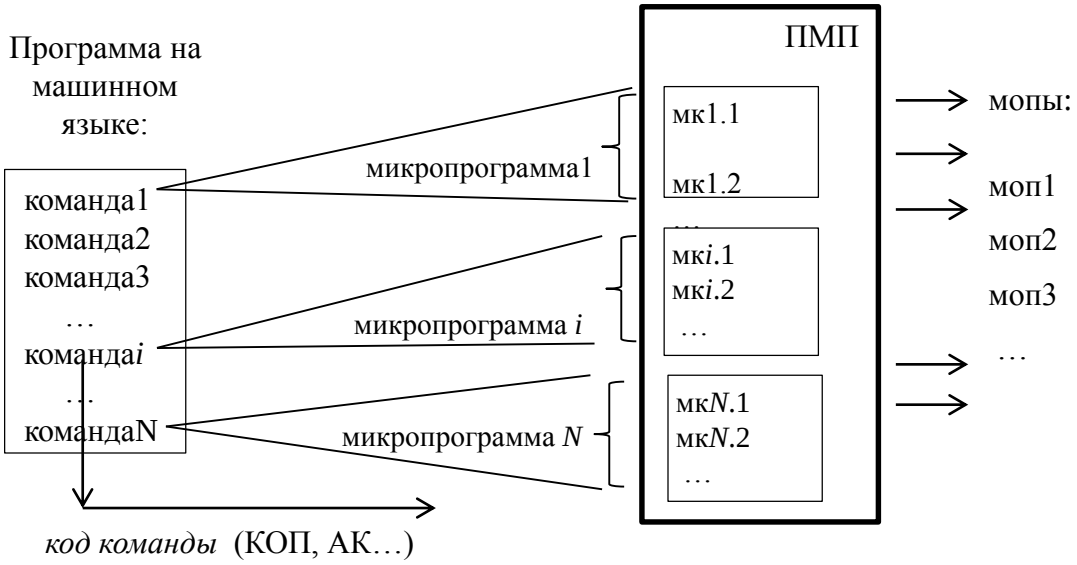


Функциональная схема

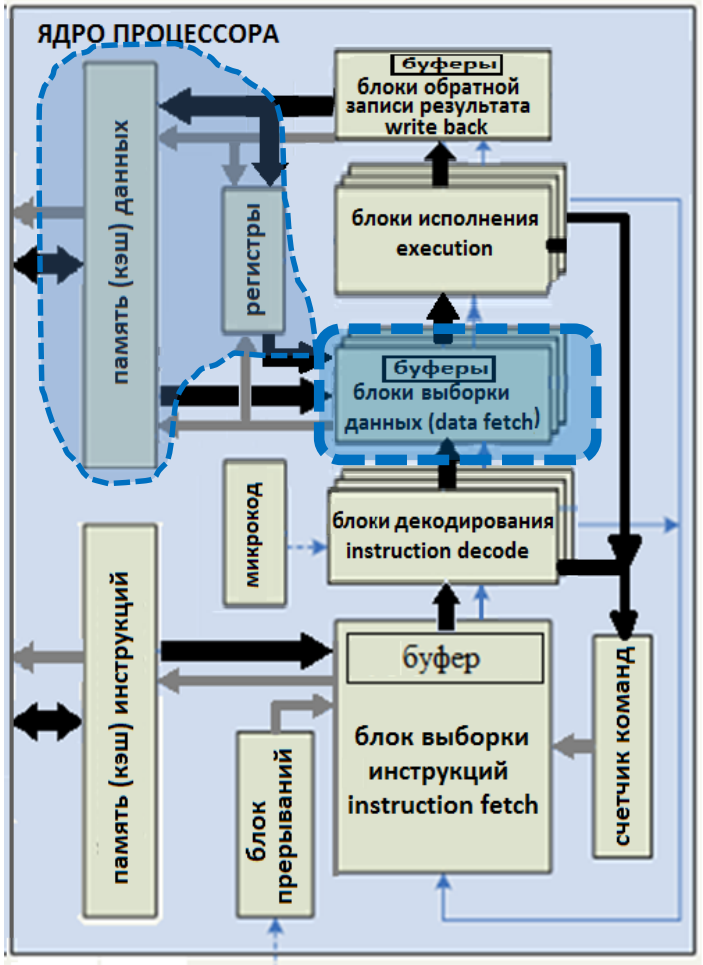
БЛОКИ ДЕКОДИРОВАНИЯ КОМАНД (Фаза2)

- *Расшифровка кодов операции, префиксов, адресов, кодов непосредственных операндов*
- *Формирование кодов микроопераций (моп) или управляющих сигналов (УС)*

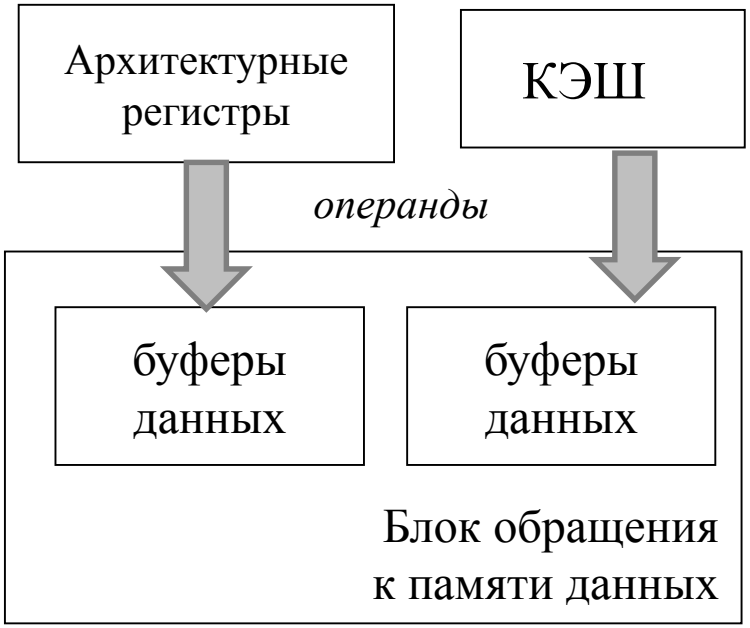




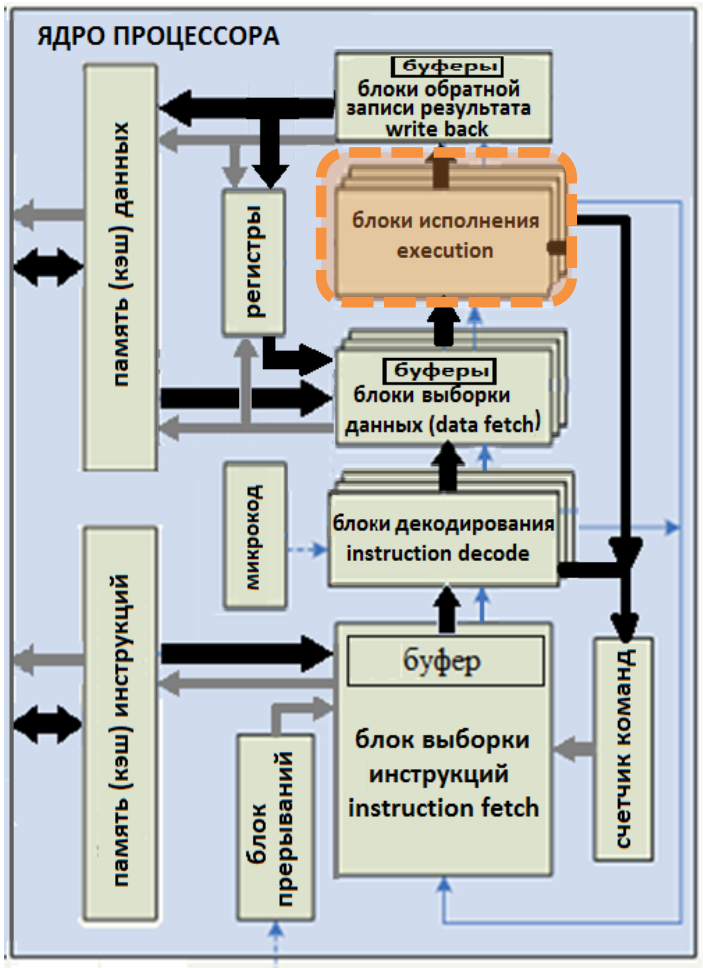
БЛОКИ ВЫБОРКИ ОПЕРАНДОВ (Фаза3)



- *Формирование адресов операндов (на основании полей АК в команде)*
- *Выборка операндов из ОП во внутренние буферы*
- *Размещение данных из архитектурных регистров во внутренних регистрах*



БЛОКИ ИСПОЛНЕНИЯ ОПЕРАЦИЙ(Фаза4)



- *исполнение микроопераций*
- *изменение состояния задачи по результату*
- *сохранение результата во внутреннем буфере*

Множество различных исполнительных блоков для обработки разного типа данных:

- целых
- дробных
- векторов
- адресов

По функциональному назначению процессора

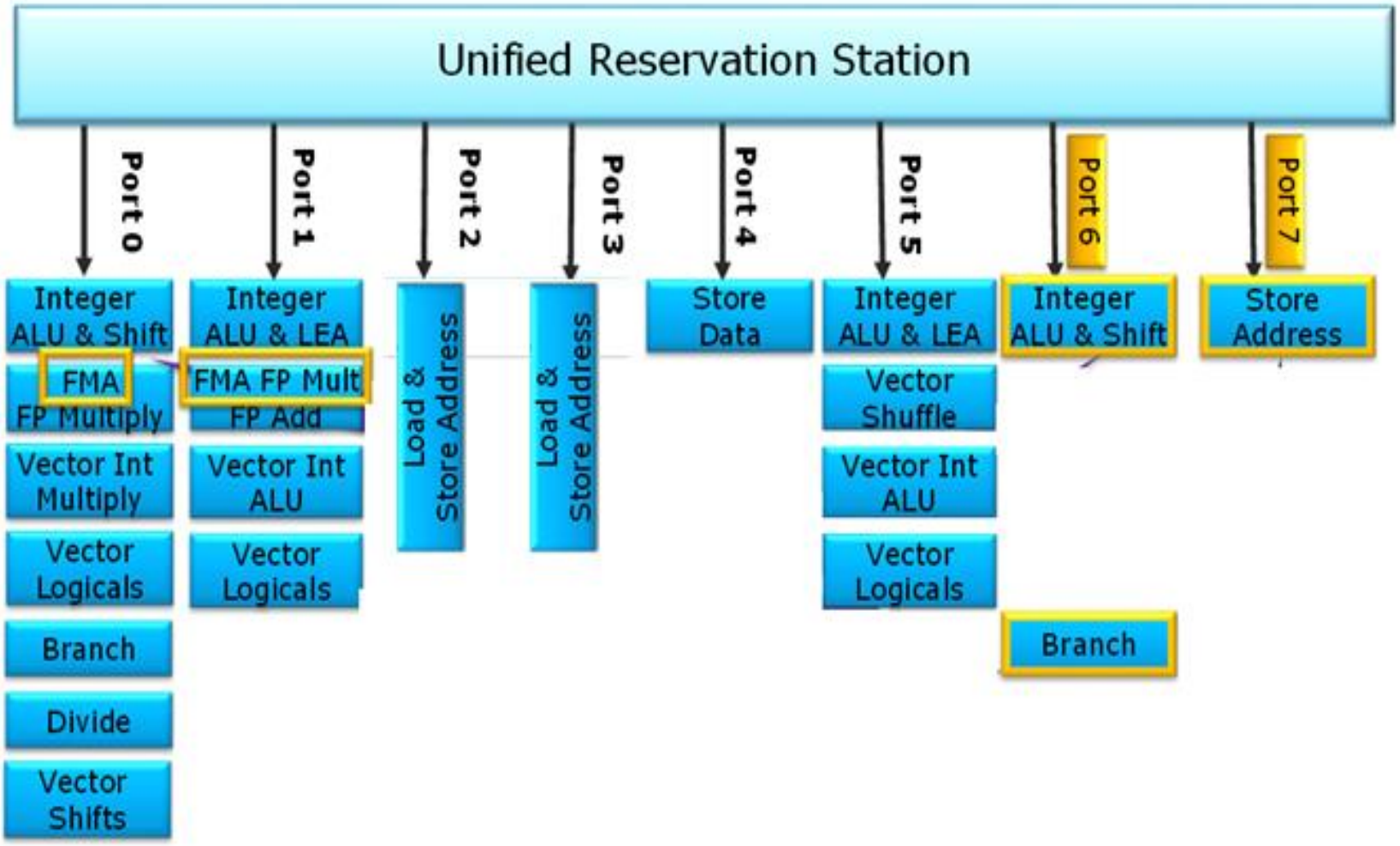
БЛОКИ ИСПОЛНЕНИЯ ОПЕРАЦИЙ (Фаза4)

Зависят от типа команд в наборе инструкций процессора/ВС

Группа команд общего назначения (общие для всех поколений процессоров данного типа)

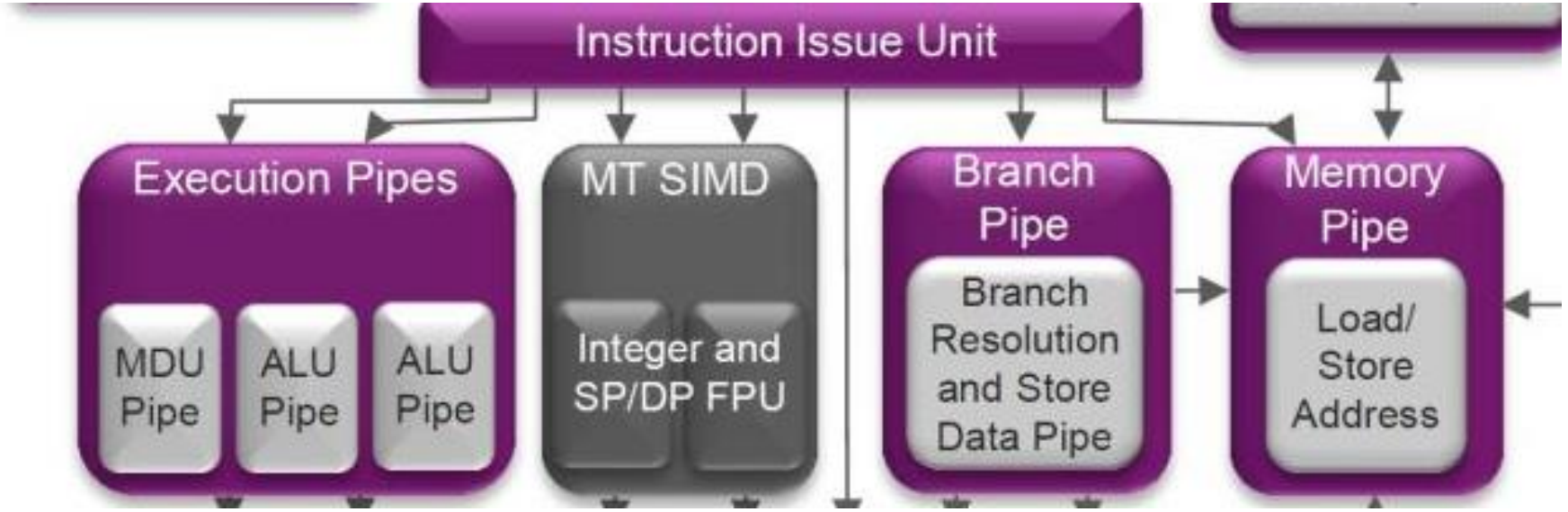
1. передачи данных (пересылки MOV, загрузки адресов LEA, LDS, LGDT…) **блок LOAD & STORE DATA**
2. обработки данных
   - арифметические команды над целыми числами с фиксированной запятой (ADD, SUB, MUL, CMP…) **блок INTEGER**
   - логические операции (AND, OR, NEG) **блок LOGICAL**
   - операции над битовыми и байтовыми полями (BTC, SETB)) **блок LOGICAL**
   - команды сдвига (SHR, ROL)) **блок SHIFT**
   - команды обработки дробных чисел с плавающей запятой (FCOMP, F2XM1,…) **блок FP, FMA, AVX**
   - команды обработки десятичных чисел (DAA, DAS,…) **блок INTEGER**
   - команды обработки строк (CMPSB, LODS,…) **блок INTEGER**
   - команды обработки векторов (MASKMOVQ, PADDB,…) **блок VECTOR**
   - команды адресной арифметики (доступ к ОП) **блок LOAD & STORE ADDRESS**
3. передачи управления (Jxxx, LOOP, CALL, RET…) **блок BRANCH**
4. команды ввода/вывода (IN, OUT) **блок INTEGER**
5. команды управления регистром флага (STC, CLD, LAHF) **блок INTEGER**
6. дополнительные команды (например, команды управления режимами работы процессора (WAIT), управления кэшированием (INVD), пустая операция (NOP). **блок INTEGER**
7. Группы команд специфические для отдельных поколений процессоров (расширения SSE/SSE2/SSE3, MMX…) спец. блоки

БЛОКИ ИСПОЛНЕНИЯ ОПЕРАЦИЙ (Фаза4)



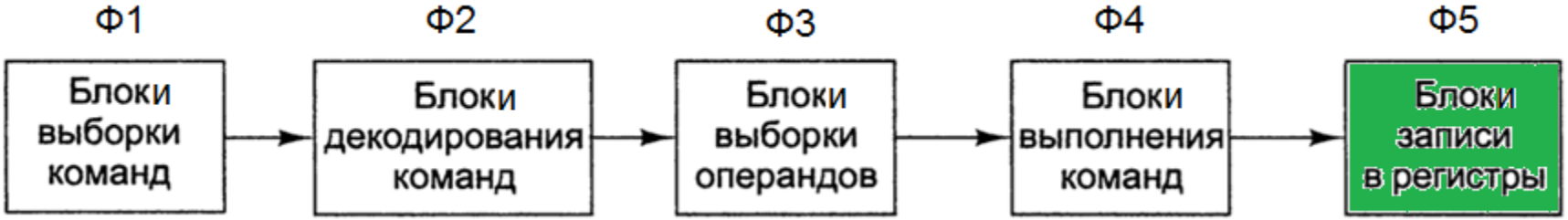
8 исполнительных устройств в микроархитектуре Intel Haswell/Skylake

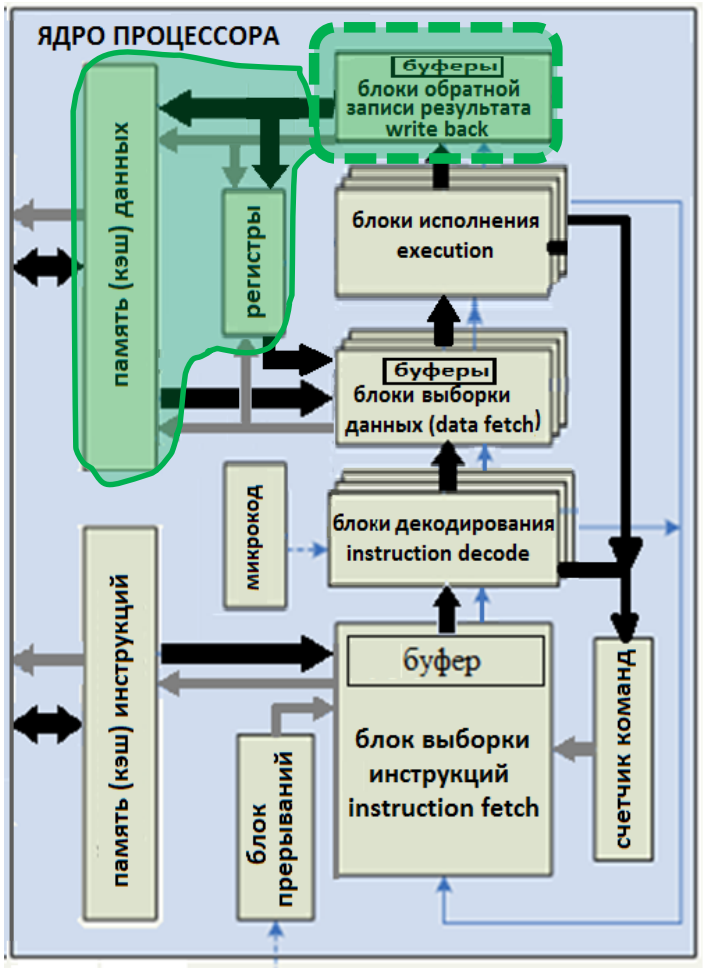
БЛОКИ ИСПОЛНЕНИЯ ОПЕРАЦИЙ (Фаза4)



5 исполнительных устройств в микроархитектуре MIPS I6400

БЛОКИ ЗАПИСИ РЕЗУЛЬТАТОВ (Фаза5)

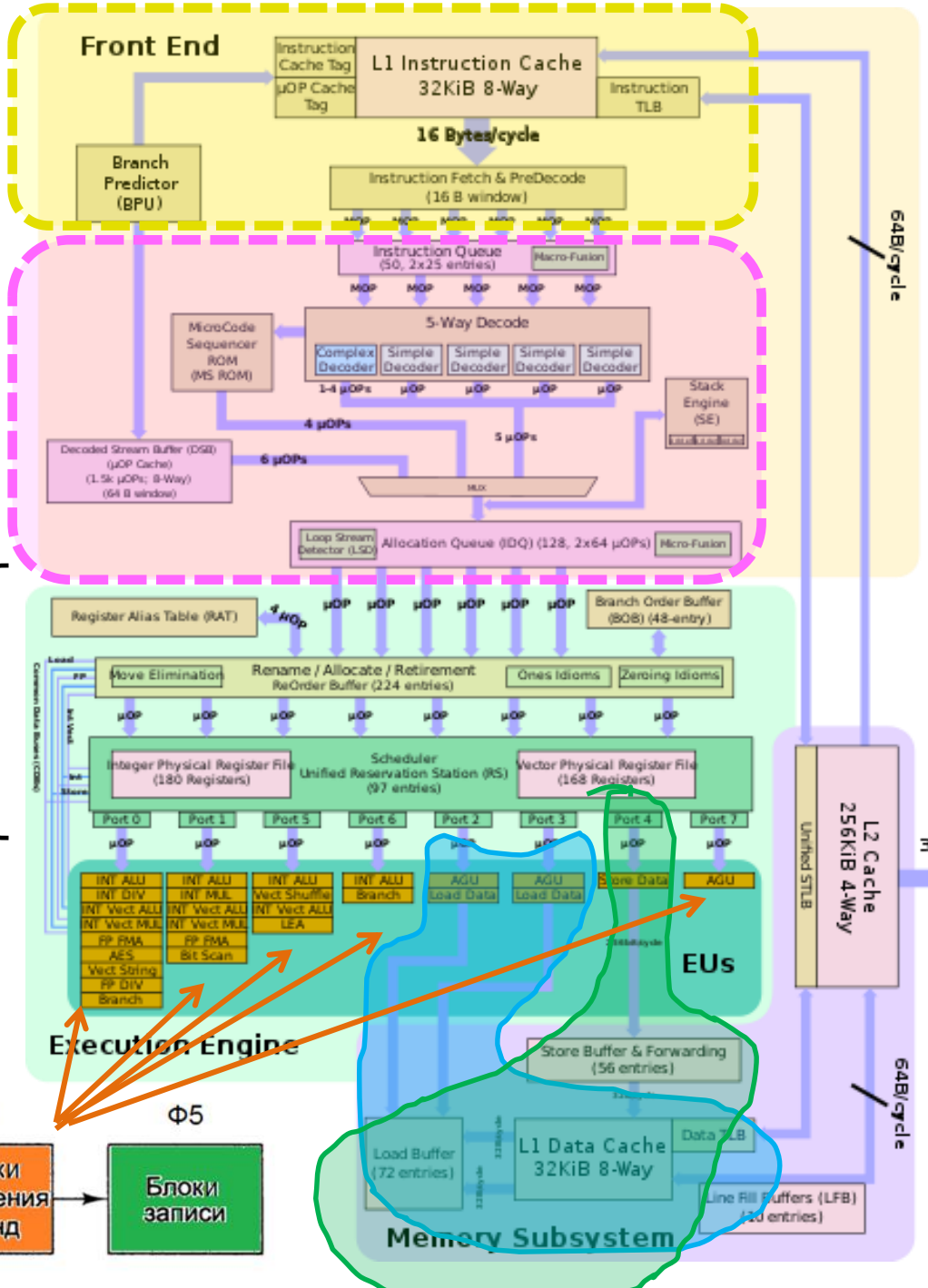


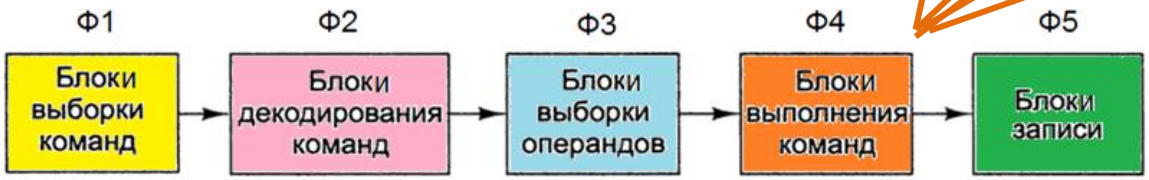


- *Сохранение результата операции в памяти /регистре*

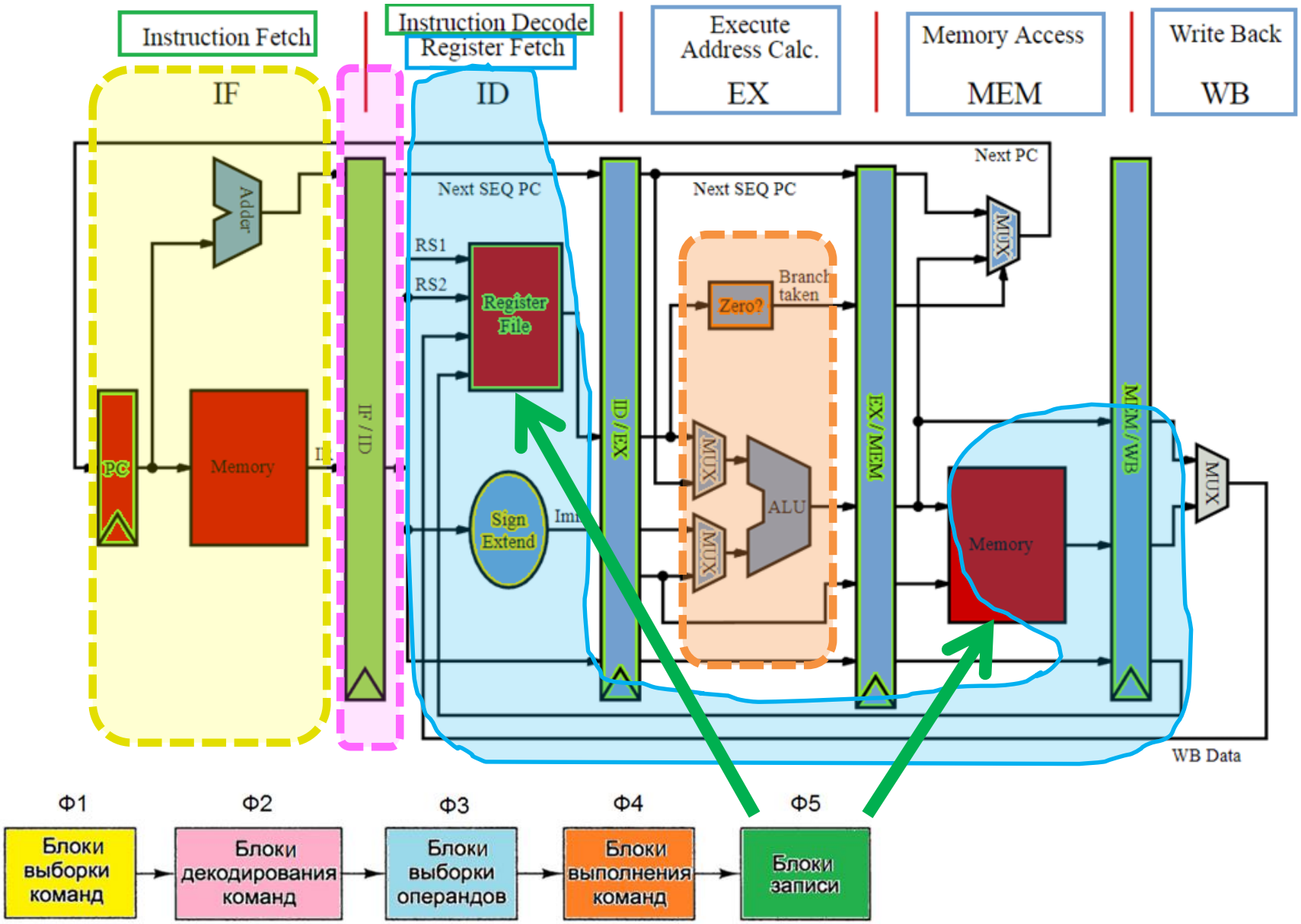
Intel Skylake

Дополнительные интеллектуальные блоки управления и диспетчеризации



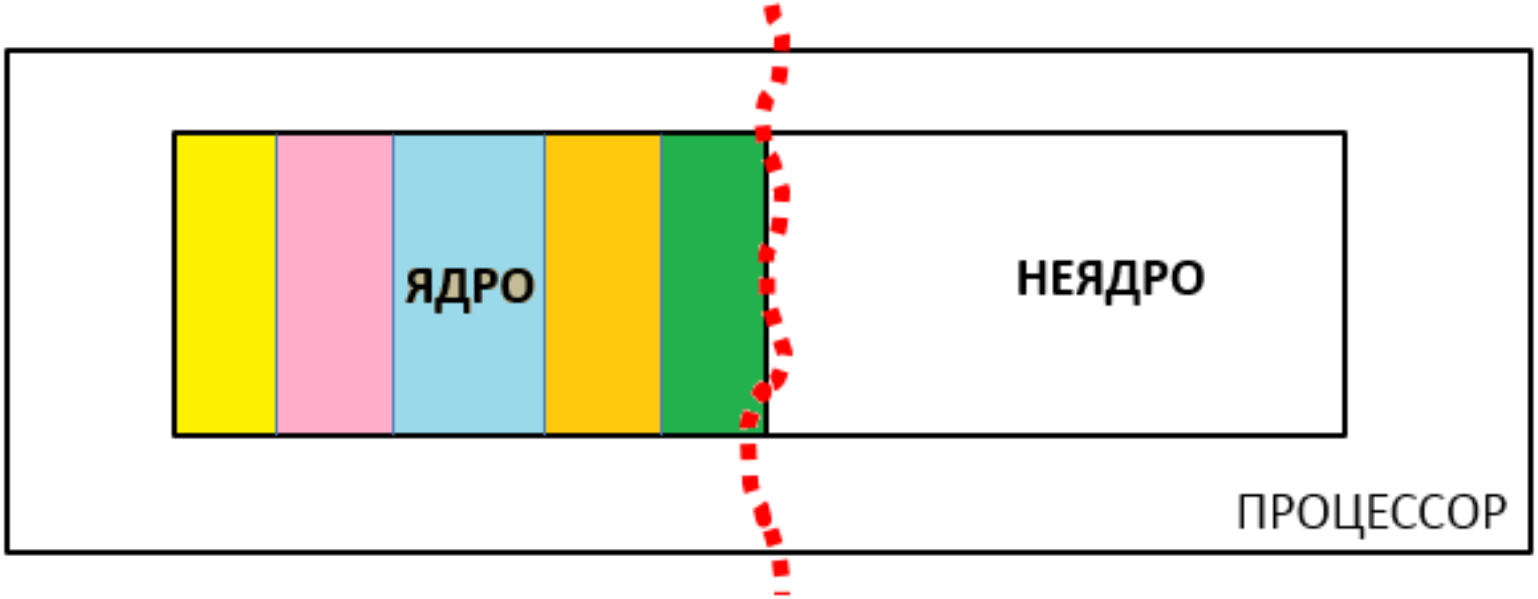


MIPS


Instruction Fetch
Instruction Decode
Register Fetch
Execute
Address Calc.
Memory Access
Write Back
IF
ID
EX
MEM
WB
Next PC
Next SEQ PC
Next SEQ PC
Adder
RS1
RS2
Register
File
Branch
taken
Zero?
MUX
PC
Memory
IF/ID
Sign
Extend
ID/EX
ALU
EX/MEM
Memory
MEM/WB
WB Data
Ф1
Ф2
Ф3
Ф4
Ф5
Блоки выборки команд
Блоки декодирования команд
Блоки выборки операндов
Блоки выполнения команд
Блоки записи

Состав блоков процессора («ядро» + «неядро»)

Решение на одном кристалле (SoC=СнК)



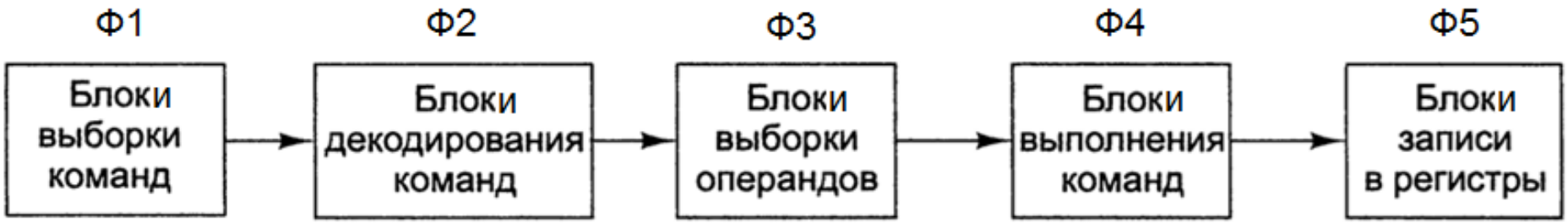
ЯДРО =

= средства выполнения команд програмы+локальные КЭШ

НЕЯДРО

= средства для выполнения задач, общих для всей ВС: сетевое взаимодействие, мультимедиа, южный/северный мост, контроллеры интерфейсов и УВВ, спецпроцессоры (DSP, безопасность, …) + Память (общая КЭШ LL верхнего уровня)

Порядок исполнения потока команд
Последовательная обработка



Загрузка блоков ЦП

Ф1
Ф2
Ф3
Ф4
Ф5

Время выполнения команды (такты)

Блок выборки команды	Блок декодирования команд	Блок выборки операндов	Блок исполнения операндов	Блок записи результатов
Ф1				
	Ф2			
		Ф3		
			Ф4	
				Ф5

Последовательная обработка

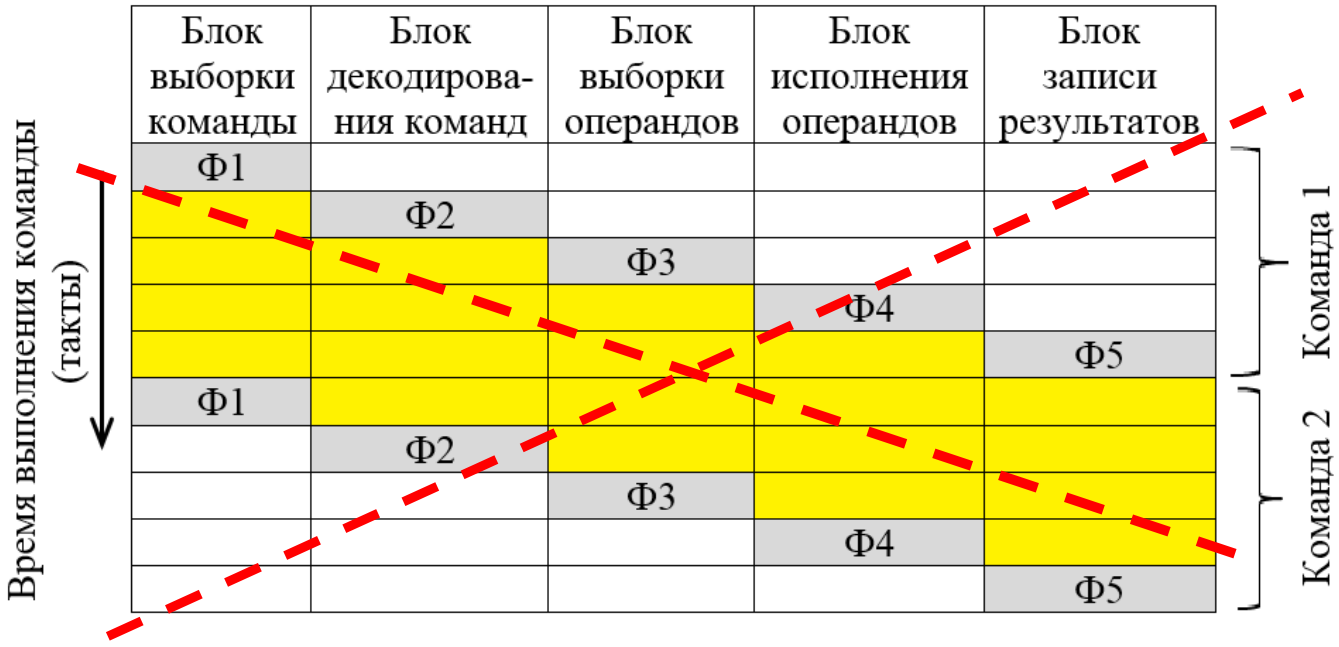
Загрузка блоков ЦП

Время выполнения команды (такты)

Блок выборки команды	Блок декодирования команд	Блок выборки операндов	Блок исполнения операндов	Блок записи результатов
Ф1				
	Ф2			
		Ф3		
			Ф4	
				Ф5
Ф1				
	Ф2			
		Ф3		
			Ф4	
				Ф5

Команда 1

Команда 2



Последовательная обработка команд

с простоями блоков

Загрузка блоков ЦП

Блок выборки команды	Блок декодирования команд	Блок выборки операндов	Блок исполнения операций	Блок записи результатов
Ф1 К1				
Ф1 К2	Ф2 К1			
Ф1 К3	Ф2 К2	Ф3 К1		
Ф1 К4	Ф2 К3	Ф3 К2	Ф4 К1	
Ф1 К5	Ф2 К4	Ф3 К3	Ф4 К2	Ф5 К1
	Ф2 К5	Ф3 К4	Ф4 К3	Ф5 К2
		Ф3 К4	Ф4 К5	Ф5 К3
			Ф4 К5	Ф5 К4

Время выполнения команды (такты)

Команда 1

Команда 2

Параллельная обработка команд без простоев блоков ЦП

конвейер команд

КОНВЕЙЕРНОЕ ИСПОЛНЕНИЕ КОМАНД

Это параллельная обработка процессором нескольких команд, при которой следующая команда программы запускается на исполнение, не дожидаясь окончания исполнения предыдущей команды, и во времени совмещается выполнение разных фаз разных команд

Загрузка блоков ЦП

Время выполнения команды (такты)

	Блок выборки команды	Блок декодирования команд	Блок выборки операндов	Блок исполнения операций	Блок записи результатов
1	Ф1 К1				
2	Ф1 К2	Ф2 К1			
3	Ф1 К3	Ф2 К2	Ф3 К1		
4	Ф1 К4	Ф2 К3	Ф3 К2	Ф4 К1	
5	Ф1 К5	Ф2 К4	Ф3 К3	Ф4 К2	Ф5 К1
6		Ф2 К5	Ф3 К4	Ф4 К3	Ф5 К2
7			Ф3 К4	Ф4 К5	Ф5 К3
8				Ф4 К5	Ф5 К4

На такте 5 совмещается пять разных фаз (Ф1-Ф2-…-Ф5) пяти разных команд (К1, К2, .., К5).