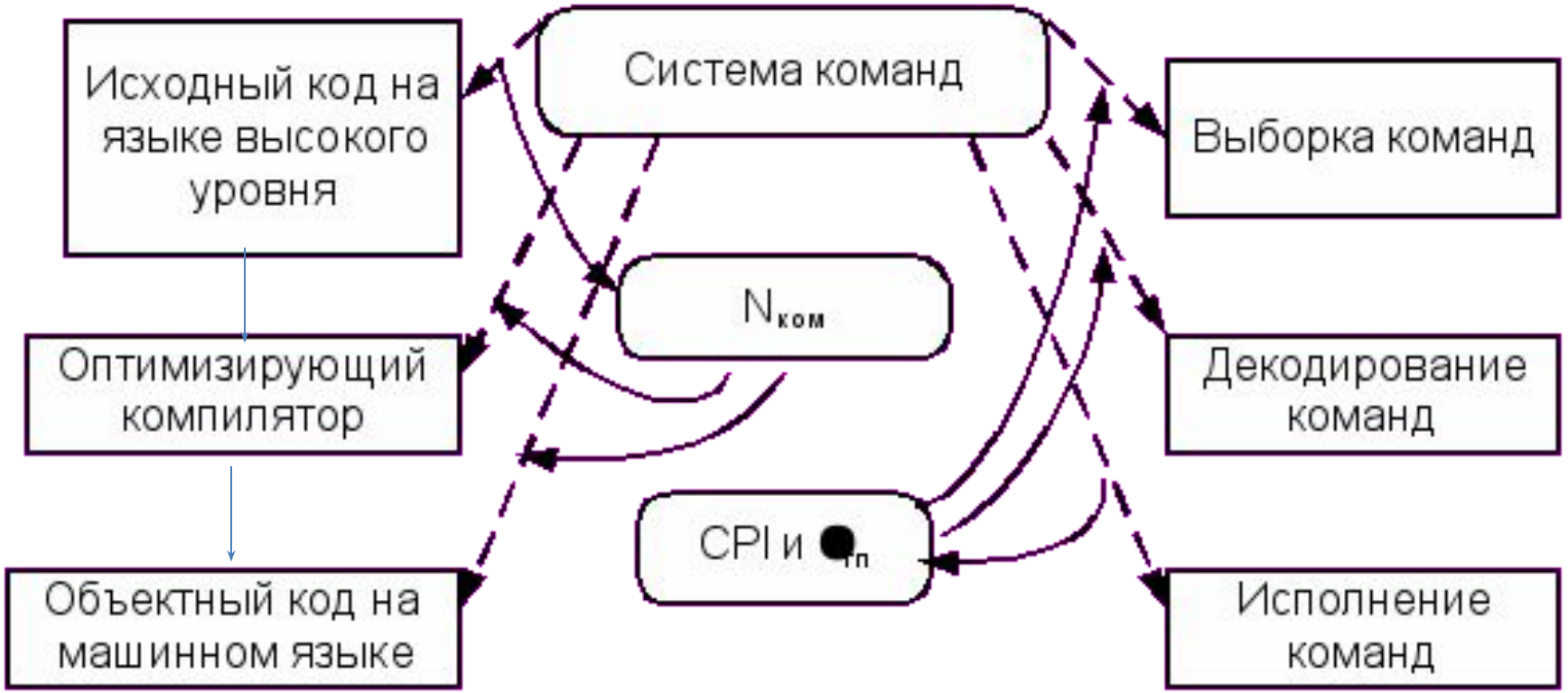
Архитектура системы команд

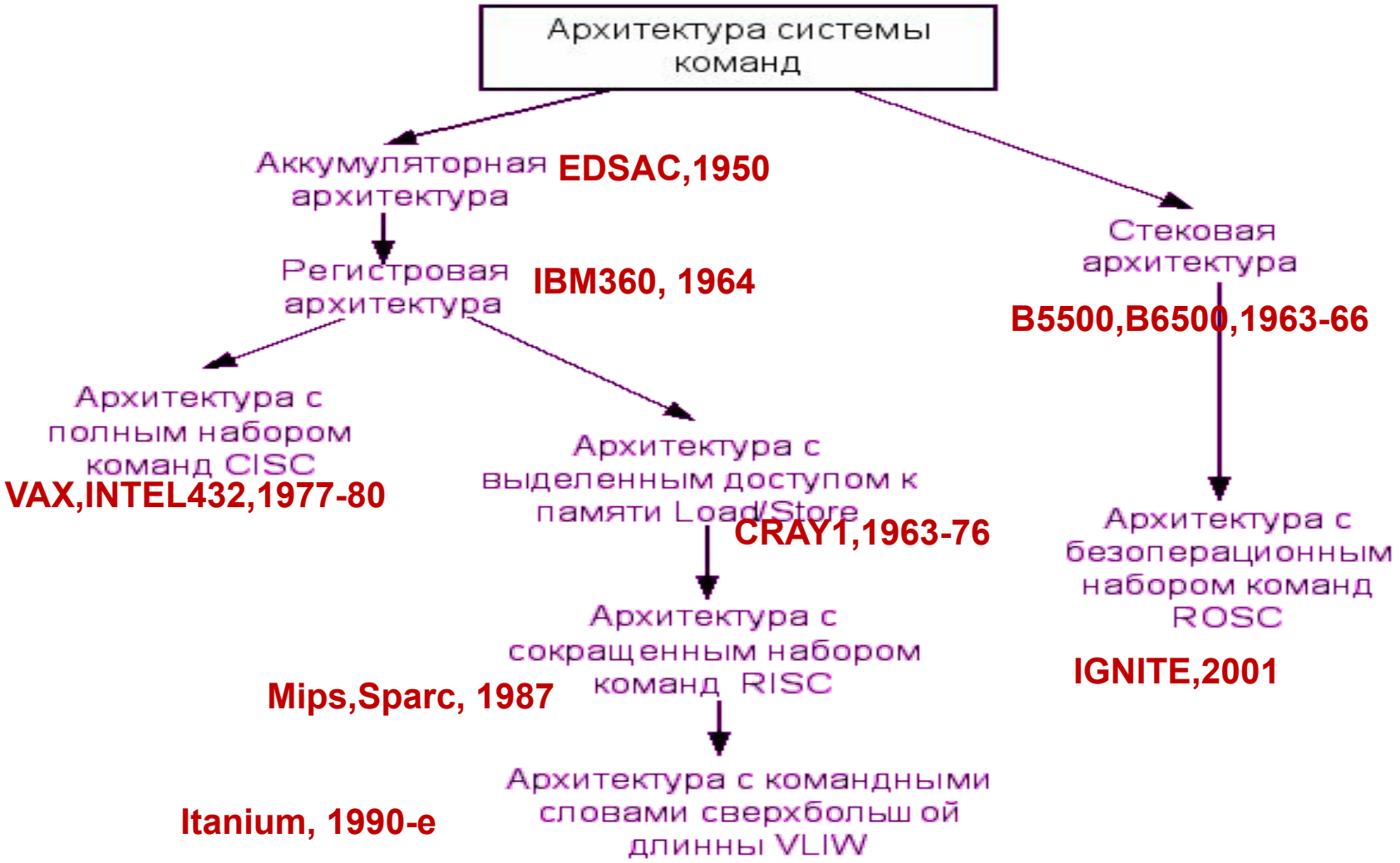
Время выполнения программы $T_{выч}$ можно определить:

- через число команд в программе $N_{ком}$,
- среднее количество тактов процессора, приходящихся на одну команду CPI
- длительность тактового периода $t_{тп}$:

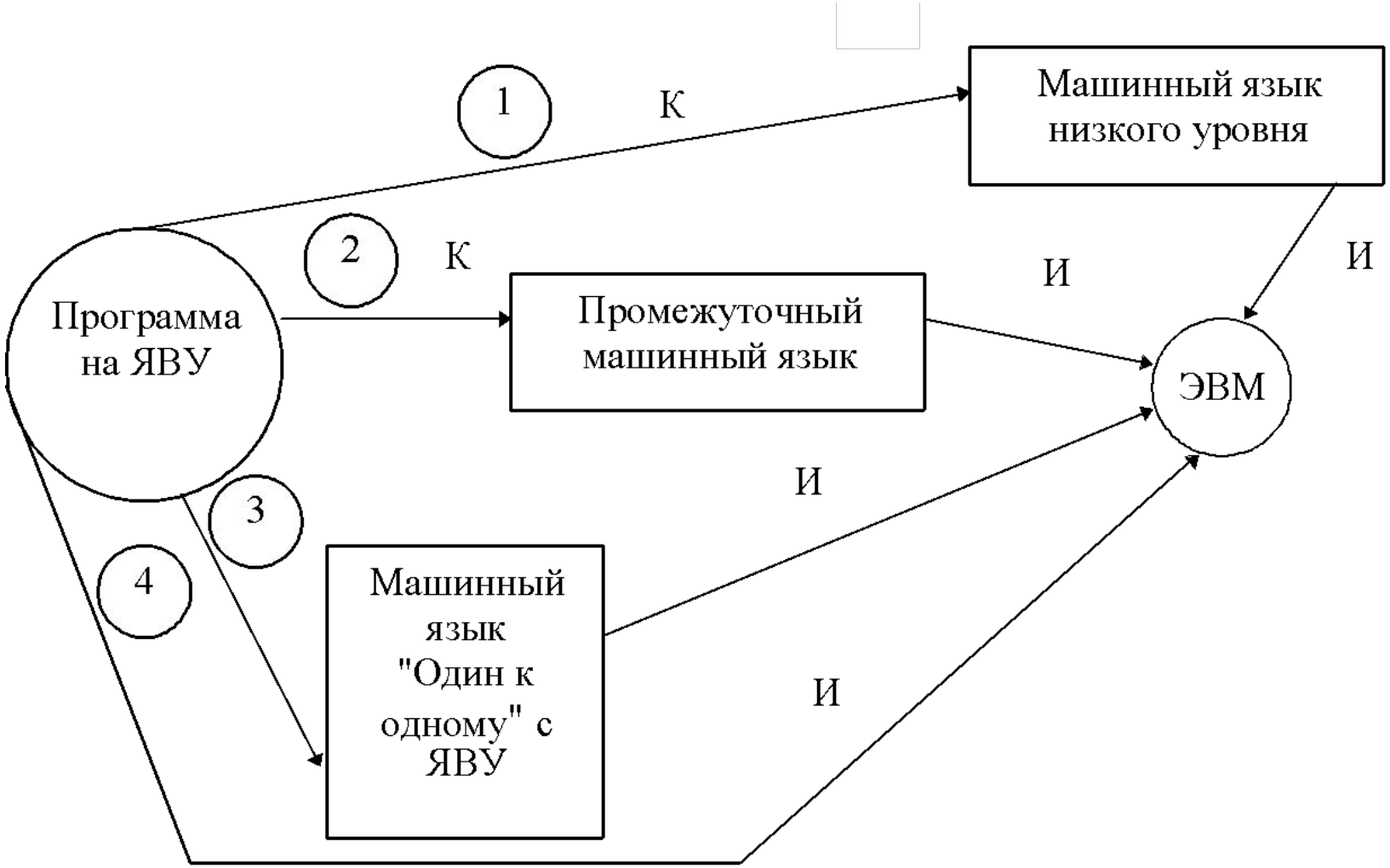
$T_{выч} = N_{ком} * CPI * t_{тп}$.



Классификация архитектур системы команд



Соотношение программ на ЯВУ и машинном языке



Обозначения принятые на рисунке «Соотношение программ на ЯВУ и машинном языке »

1. это традиционный подход. После компилирования программа переводится на машинный язык, а затем интерпретируется машиной;
2. компиляция идет на машинный язык более высокого уровня, сокращая тем самым семантический разрыв между ЯВУ и машиной;
3. здесь ЯВУ можно рассматривать как язык ассемблера, т.е. имеется взаимно однозначное соответствие между типами операторов и знаков операций ЯВУ с командами машинного языка. Здесь идет ассемблирование, а не компилирование, во время которого удаляются комментарии и пробелы в исходной программе, преобразуются разделители, ключевые слова и знаки операций в машинные коды, имена – в адреса полей памяти. Таким образом, многих привычных функций компилятора здесь нет. Остальная привязка программы к ЭВМ происходит перед выполнением программы;
4. здесь машинный язык является ЯВУ и идет процесс интерпретации программы на компьютере

Классификация по составу и сложности команд

Подходы к преодолению семантического разрыва

- Архитектура с полным набором команд: CISC (Complex Instruction Set Computer);
- Архитектура с сокращенным набором команд: RISC (Reduced Instruction Set Computer);
- Архитектура с командными словами сверхбольшой длины: VLIW (Very Long Instruction Word).

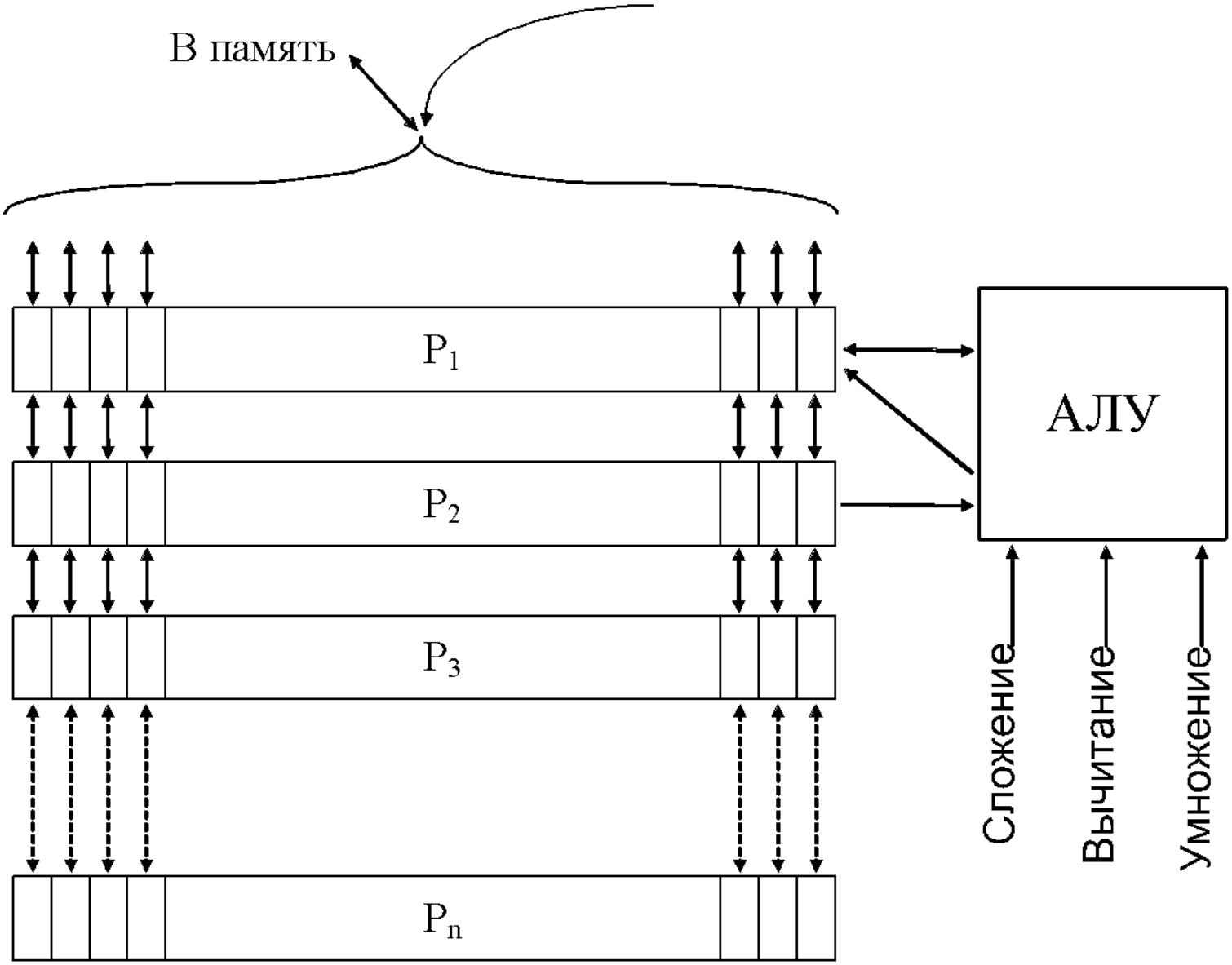
Сравнительная оценка

Характеристика	CISC	RISC	VLIW
Длина команды	Варьируется	Единая	Единая
Расположение полей в команде	Варьируется	Неизменное	Неизменное
Количество регистров	Несколько (часто специализированных)	Много регистров общего назначения	Много регистров общего назначения
Доступ к памяти	Может выполняться как часть команд различных типов	Выполняется только специальными командами	Выполняется только специальными командами

Классификация по месту хранения операндов

- Стековая
- Аккумуляторная
- Регистровая
- С выделенным доступом к памяти

Стековая организация регистровой памяти процессора



Программа решения математической задачи на ЭВМ со стековой организацией памяти

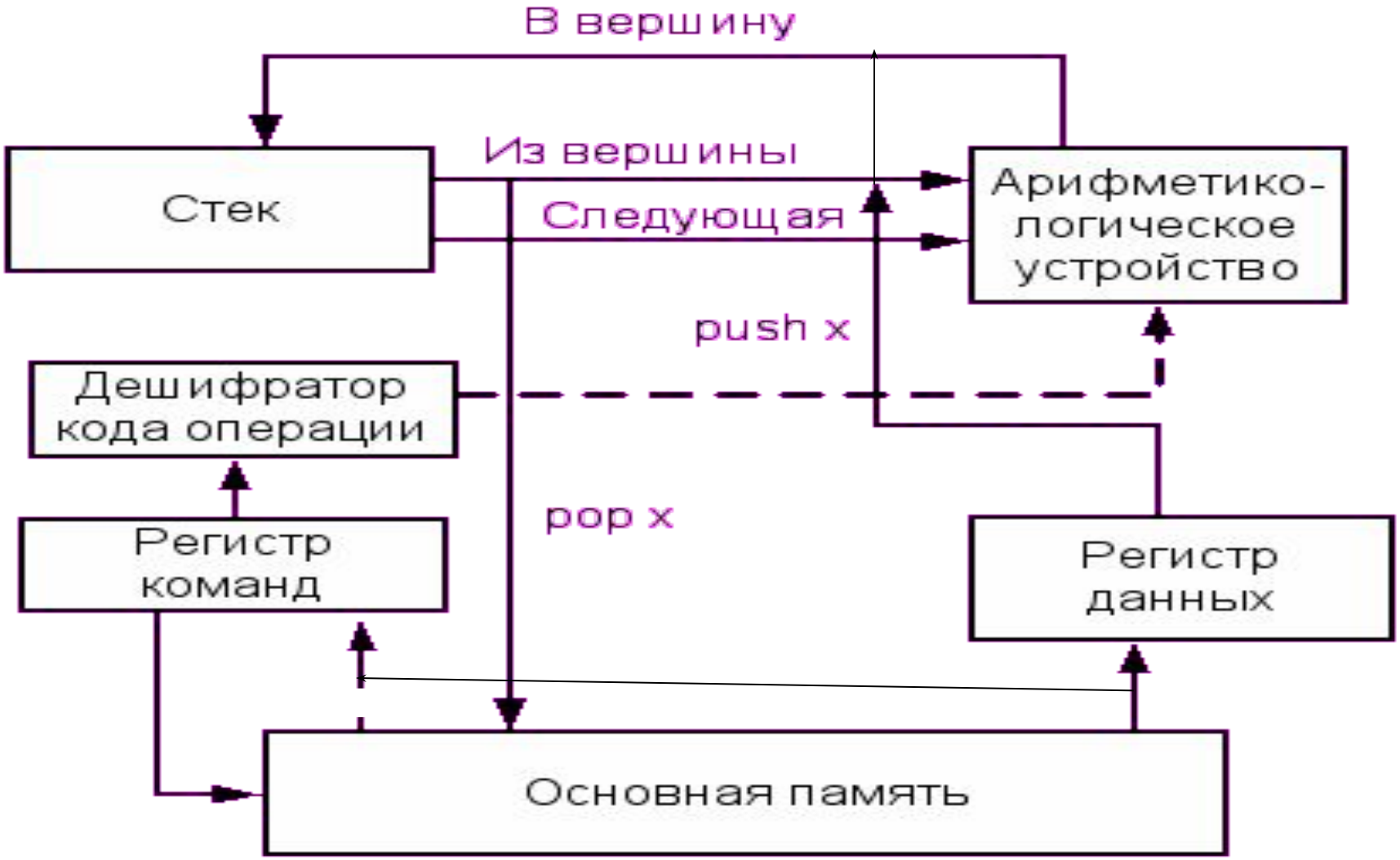
№ п/п	Команда	P_1	P_2	P_3	P_4
1	2	2	3	4	5
1	Вызов **b**	b			
2	Дублирование	b	B		
3	Вызов **c**	c	B	B	
4	Сложение	b+c	B		
5	Реверсирование	b	b+c		
6	Дублирование	b	B	b+c	
7	Умножение	b^2	b+c		
8	Вызов **a**	a	b^2	b+c	
9	Дублирование	a	A	b^2	b+c
10	Умножение	a^2	b^2	b+c	
11	Сложение	a^2+b^2	b+c		
12	Деление	$\frac{a^2+b^2}{b+c}$			

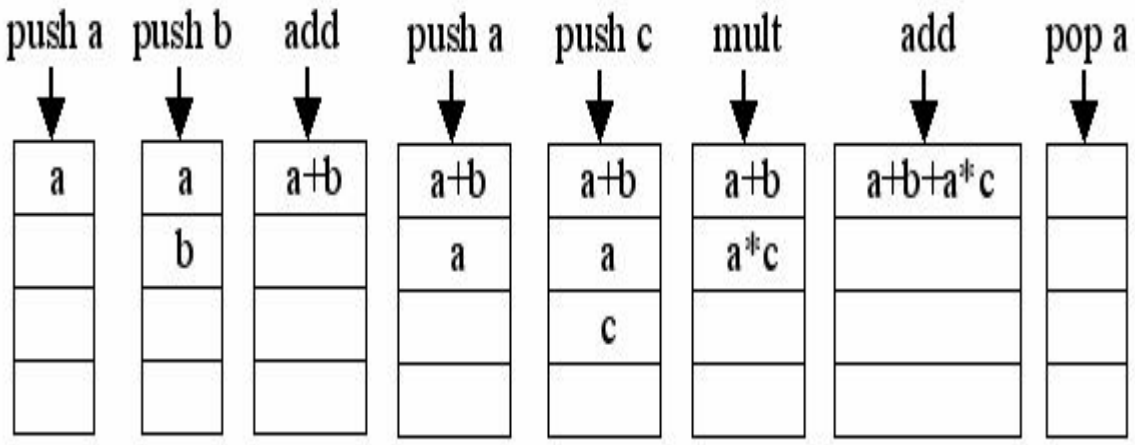
$$X = \frac{a^2 + b^2}{b + c}$$

Основные причины возникновения узких мест в компьютере

- состав, принцип работы и временные характеристики арифметико-логического устройства;
- состав, размер и временные характеристики устройств памяти;
- структура и пропускная способность коммуникационной среды;
- компилятор, создающий неэффективные коды;
- операционная система, организующая неэффективную работу с памятью, особенно медленной.

Стековая архитектура





Порядок выполнения стековых команд

Основные принципы RISC-архитектуры

- каждая команда независимо от ее типа выполняется за один машинный цикл, длительность которого должна быть максимально короткой;
- все команды должны иметь одинаковую длину и использовать минимум адресных форматов, что резко упрощает логику центрального управления процессором;
- обращение к памяти происходит только при выполнении операций записи и чтения, вся обработка данных осуществляется исключительно в регистровой структуре процессора;
- система команд должна обеспечивать поддержку языка высокого уровня. (Имеется в виду подбор системы команд, наиболее эффективной для различных языков программирования.)

Отличительные особенности CISC- и RISC-архитектур

CISC-архитектура	RISC-архитектура
Многобайтовые команды	Однобайтовые команды
Малое количество регистров	Большое количество регистров
Сложные команды	Простые команды
Одна или менее команд за один цикл процессора	Несколько команд за один цикл процессора
Традиционно одно исполнительное устройство	Несколько исполнительных устройств

Достоинства RISC-архитектуры:

1. Компактность процессора, как следствие отсутствие проблем с охлаждением;
2. Высокая скорость арифметических вычислений;
3. Наличие механизма динамического прогнозирования ветвлений;
4. Большое количество оперативных регистров;
5. Многоуровневая встроенная кэш-память;

Недостаток – проблема в обновлении регистров процессора, что привело к появлению двух методов обновления: аппаратный и программный.

Методы адресации

Метод адресации	Пример команды	Смысл команды	Использование команды
Регистровая	Add R4, R3	R4 = R4+R3	Для записи требуемого значения в регистр
Непосредственная или литерная	Add R4, #3	R4 = R4+3	Для задания констант
Базовая со смещением	Add R4, 100(R1)	R4= R4+M(100+R1)	Для обращения к локальным переменным
Косвенная регистровая	Add R4, (R1)	R4 = R4+M(R1)	Для обращения по указателю к вычисленному адресу
Индексная	Add R3, (R1+R2)	R3 = R3+M(R1+R2)	Полезна при работе с массивами: R1 – база, R3 – индекс

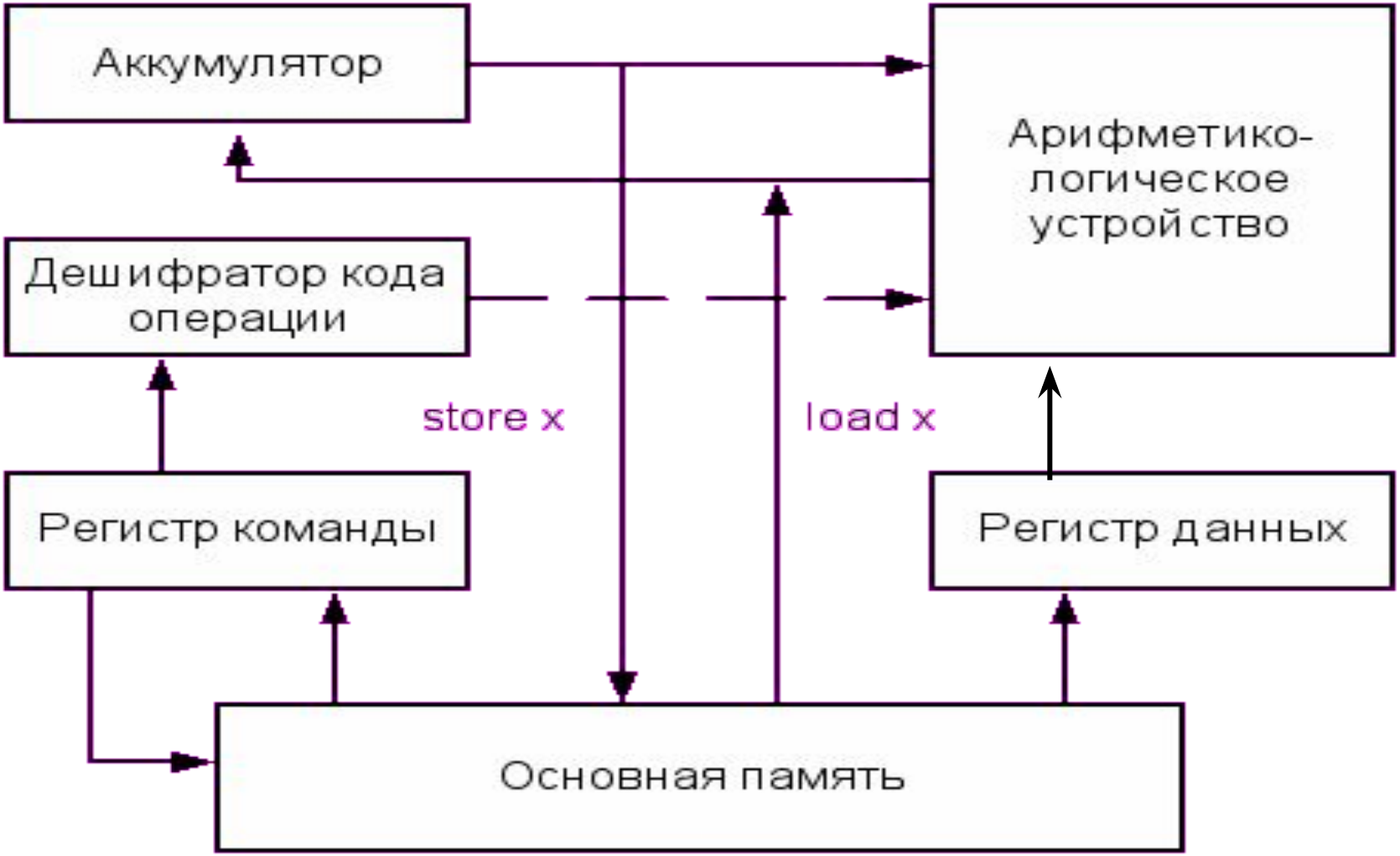
Методы адресации

Метод адресации	Пример команды	Смысл команды	Использование команды
Прямая или абсолютная	Add R1, (1000)	R1=R1+M(1000)	Полезна для обращения к статическим данным
Косвенная	Add R1, @(R3)	R1 = R1+M(M(R3))	Если R3 – адрес указателя p, то выбирается значение по этому указателю
Автоинкрементная	Add R1, (R2)+	R1 = R1+M(R2) R2 = R2+d	Полезна для прохода в цикле по массиву с шагом: R2 – начало массива. В каждом цикле R2 получает приращение d
Автодекрементная	Add R1, (R2)–	R2 = R2–d R1 = R1+M(R2)	Аналогична предыдущей. Обе могут использоваться для реализации стека
Базовая индексная со смещением и масштабированием	Add R1, 100(R2)(R3)	R1=R1+M(100)+R2 +R3*d	Для индексации массивов

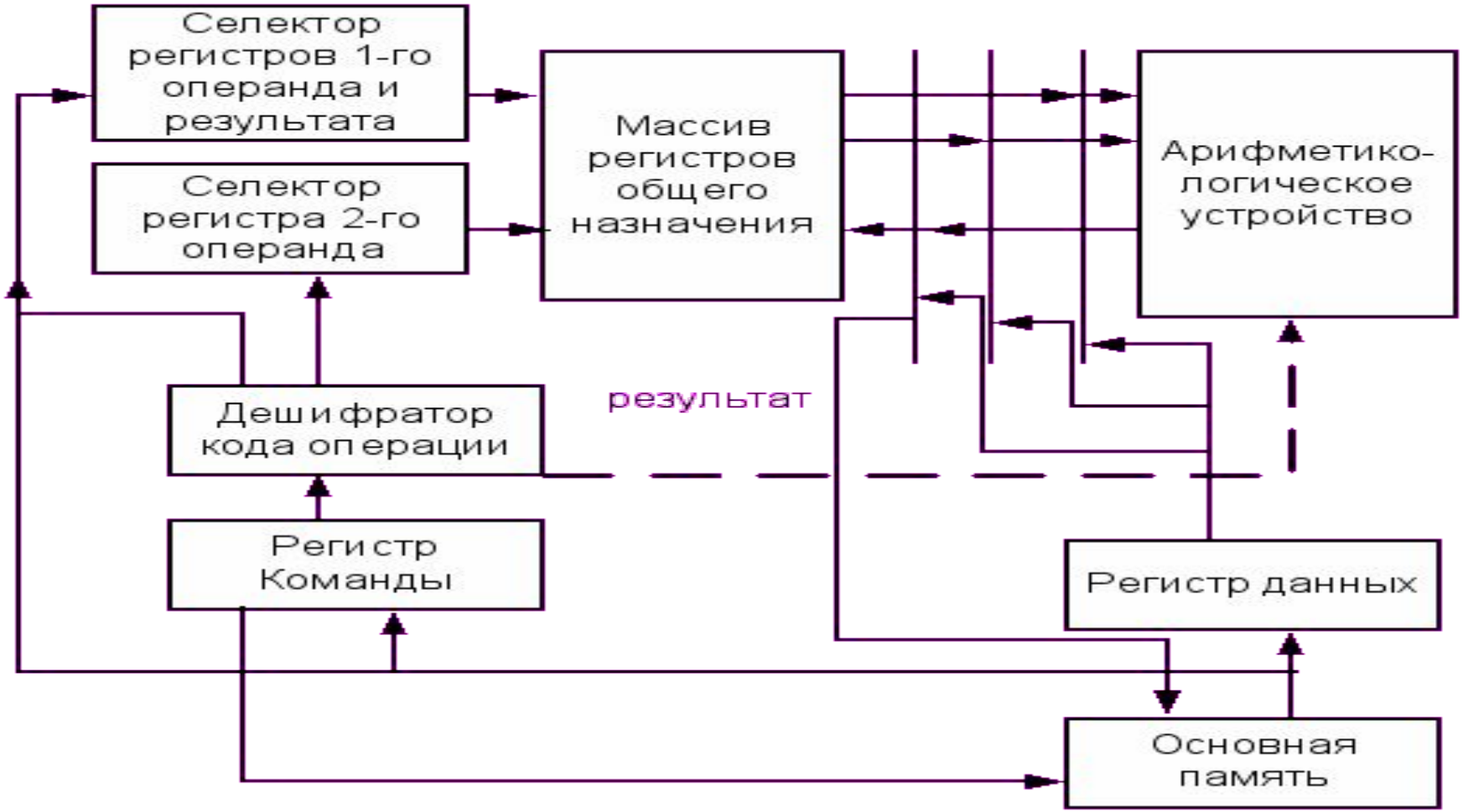
Основные типы команд

Тип операции	Примеры
Арифметические и логические	Целочисленные арифметические и логические операции: сложение, вычитание, логическое сложение, логическое умножение и т. д.
Пересылки данных	Операции загрузки/записи
Управление потоком команд	Безусловные и условные переходы, вызовы процедур и возвраты
Системные операции	Системные вызовы, команды управления виртуальной памятью и т. д.
Операции с плавающей точкой	Операции сложения, вычитания, умножения и деления над вещественными числами
Десятичные операции	Десятичное сложение, умножение, преобразование форматов и т. д.
Операции над строками	Пересылки, сравнения и поиск строк

Аккумуляторная архитектура

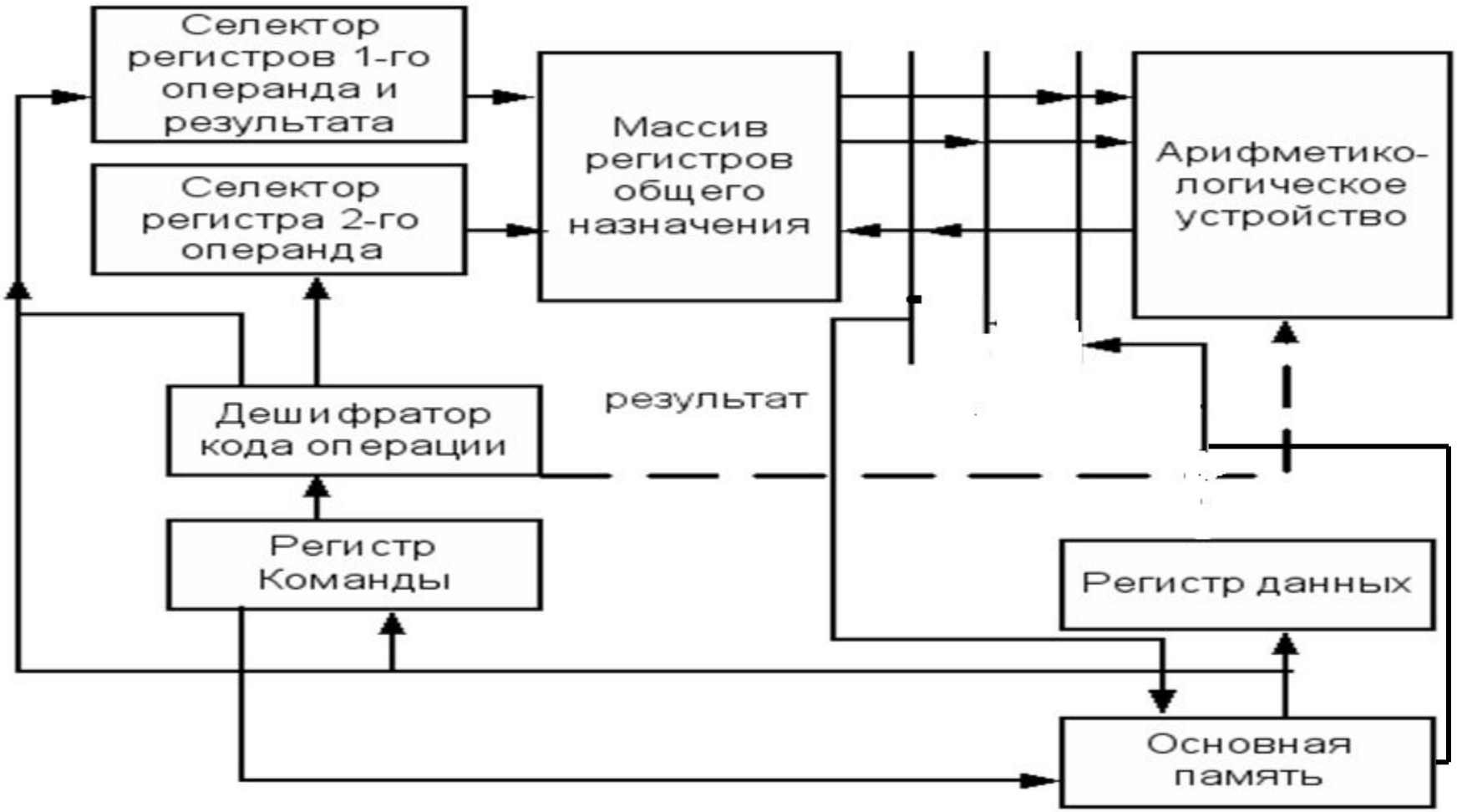


Регистровая архитектура

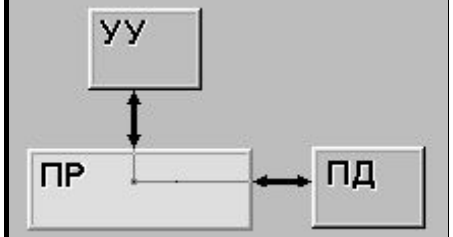
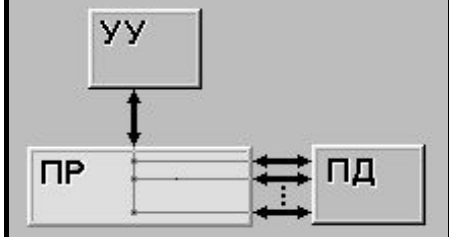
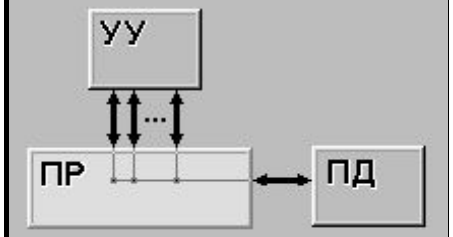
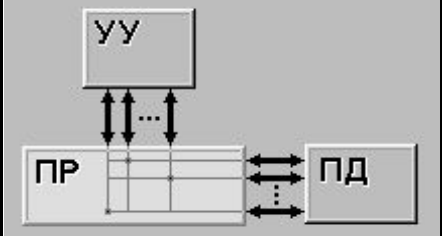


Сравнение вариантов размещения операндов

Тип команд	Достоинства	Недостатки
Регистр-регистр (0, 3)	Простота реализации; фиксированная длина команды; простые алгоритмы компиляции; одинаковый CPI для всех команд.	Большой объем объектного кода; не всегда используются все поля команды.
Регистр-память (1, 2)	Компактный объектный код; простота создания исходного кода.	Длинное поле адреса в команде; потеря одного из операндов.
Память-память (3,3)	Компактный объектный код; малая потребность в РОН.	Низкое быстродействие; разнообразие форматов команд


Селектор регистров 1-го операнда и результата
Селектор регистра 2-го операнда
Массив регистров общего назначения
Арифметико-логическое устройство
результат
Дешифратор кода операции
Регистр Команды
Регистр данных
Основная память

Классификация Флина

	**SISD** (single instruction stream / single data stream) - одиночный поток команд и одиночный поток данных.
	**SIMD** (single instruction stream / multiple data stream) - одиночный поток команд и множественный поток данных.
	**MISD** (multiple instruction stream / single data stream) - множественный поток команд и одиночный поток данных.
	**MIMD** (multiple instruction stream / multiple data stream) - множественный поток команд и множественный поток данных.