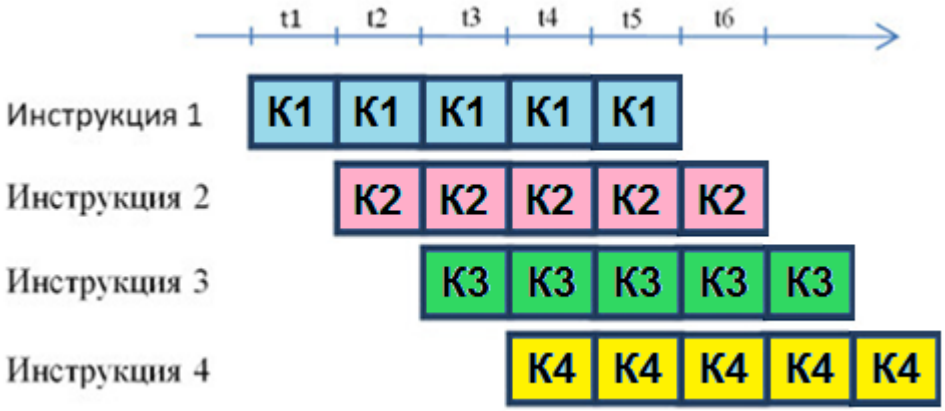
Порядок исполнения потока команд (фазы цикла исполнения команды)

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

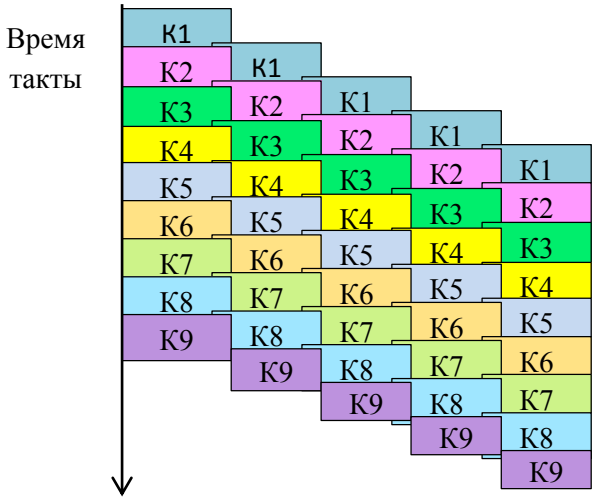
Для каждой команды все фазы должны выполняться последовательно

Последовательная обработка

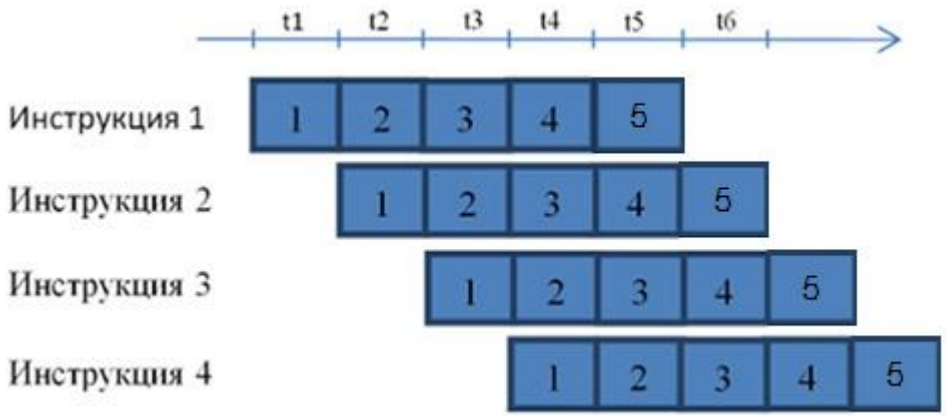
Конвейерная обработка



Блоки ЦП/ступени конвейера



Конвейерная обработка

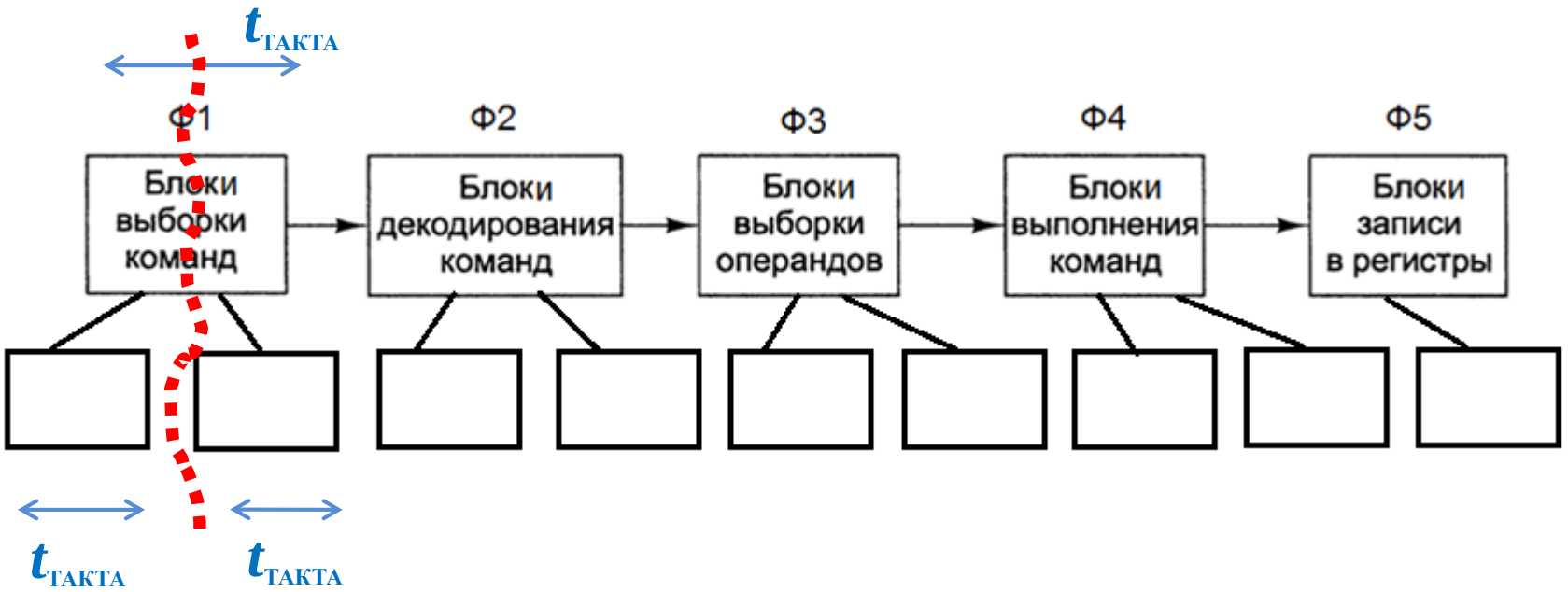


Идеальная ситуация:

$t_{\text{ступени}}$ = 1 такт

Процессоры/операции усложняются
Тактовые частоты увеличиваются

=> сложный блок не успевает



INTEL® 64 AND IA-32 ARCHITECTURES, 2-12 Vol. 1, (стр.42 (42 / 3439))

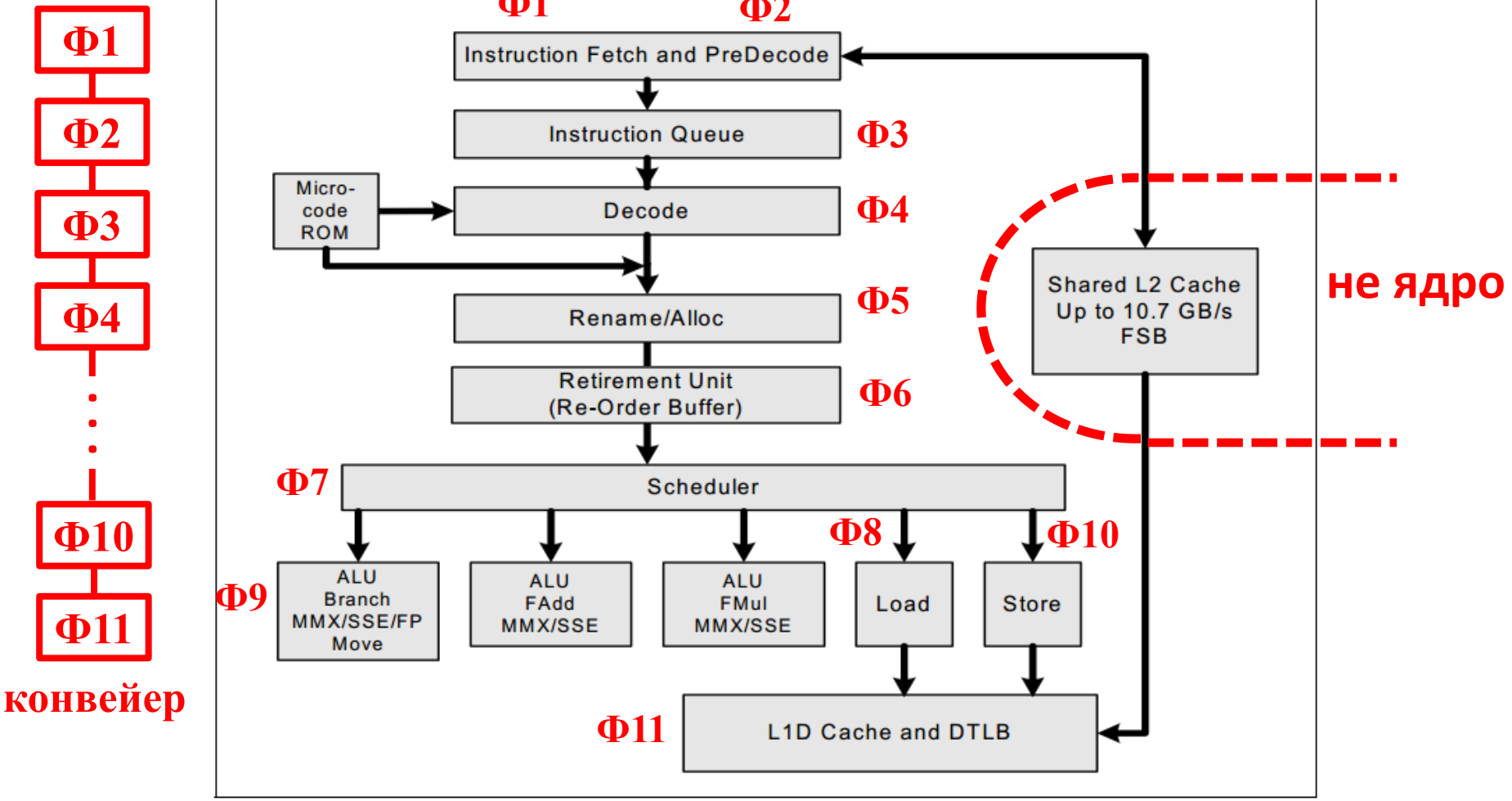
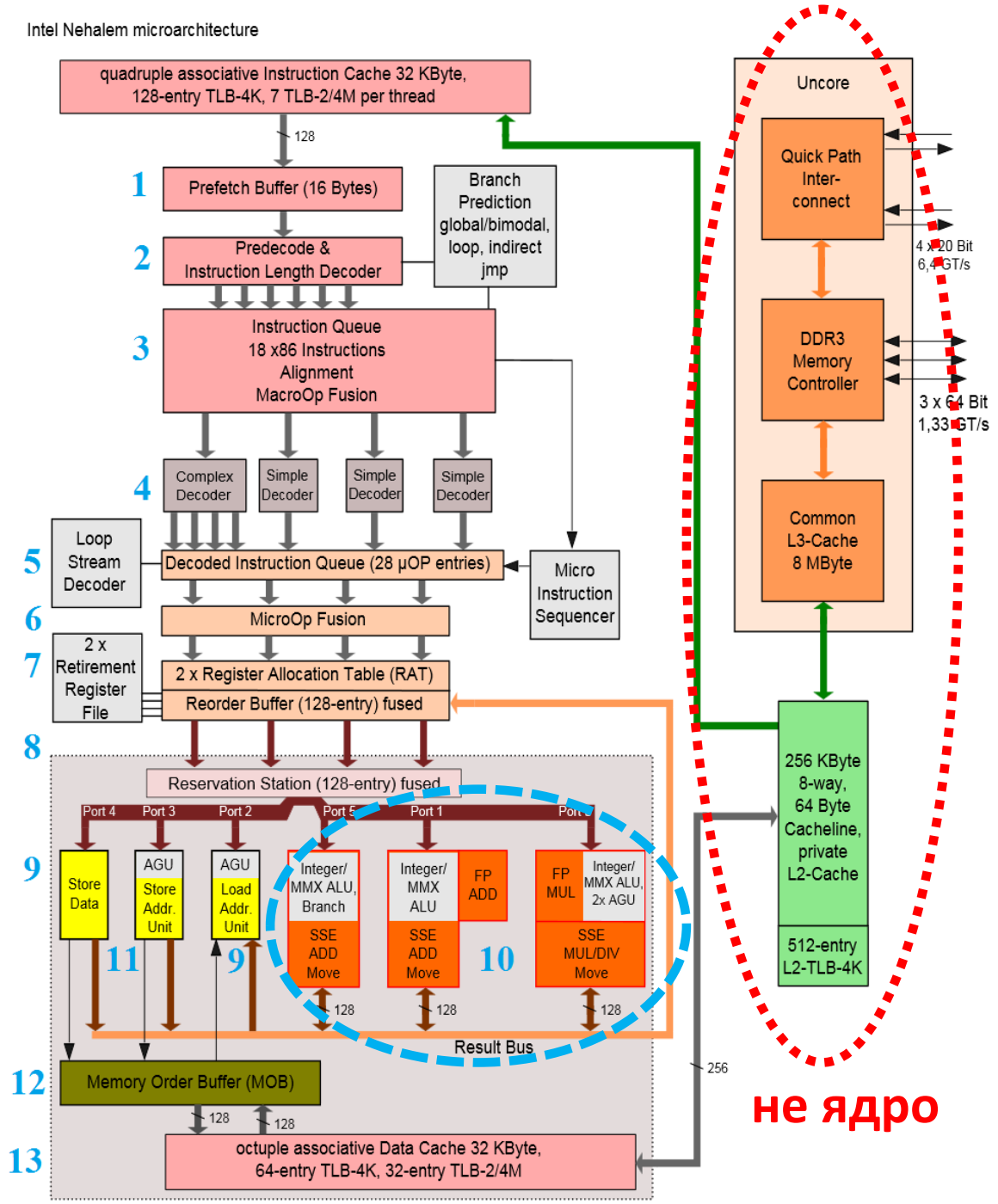


Figure 2-3. The Intel Core Microarchitecture Pipeline Functionality

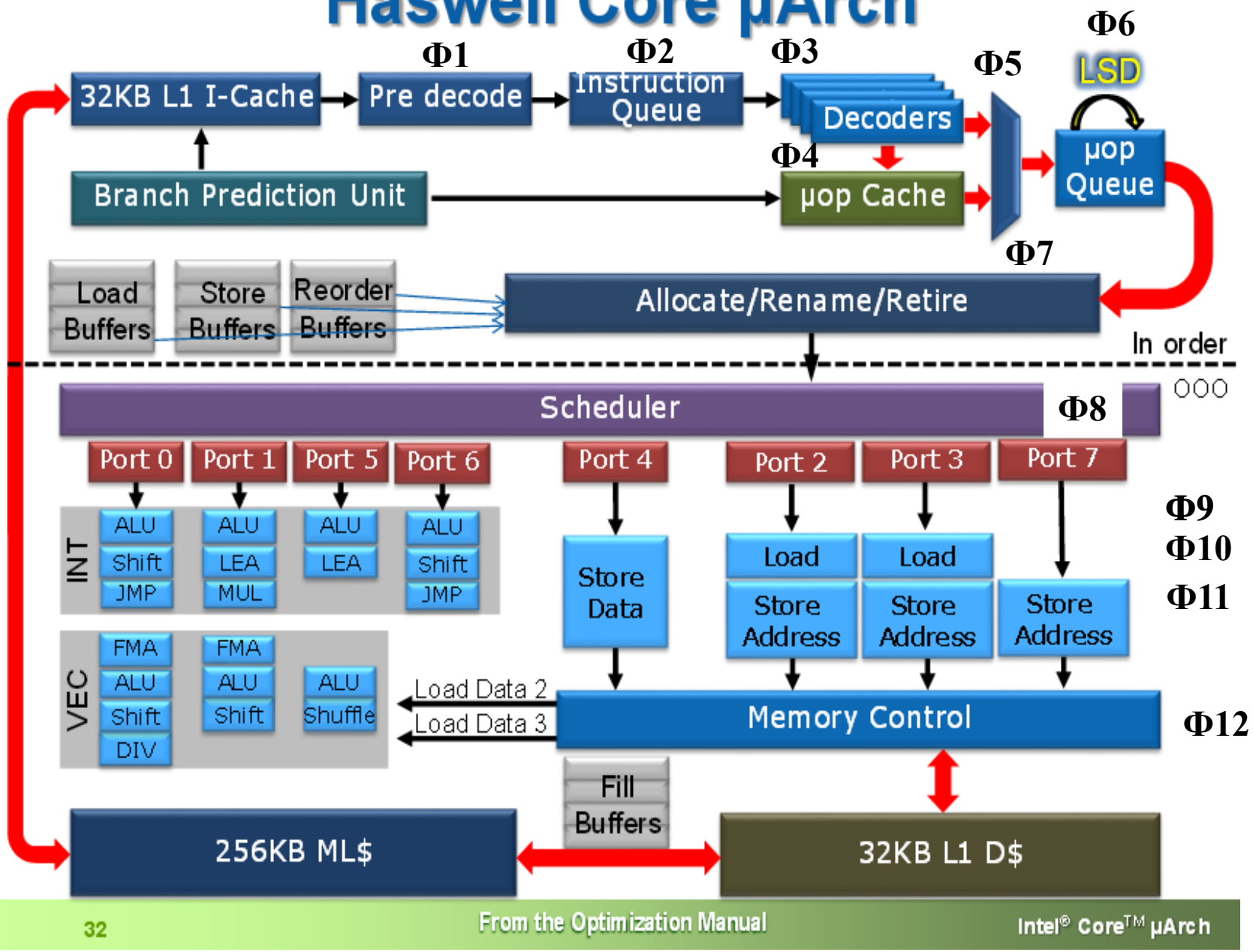
Микроархитектура конвейера ядра процессора Intel Nehalem

конвейер

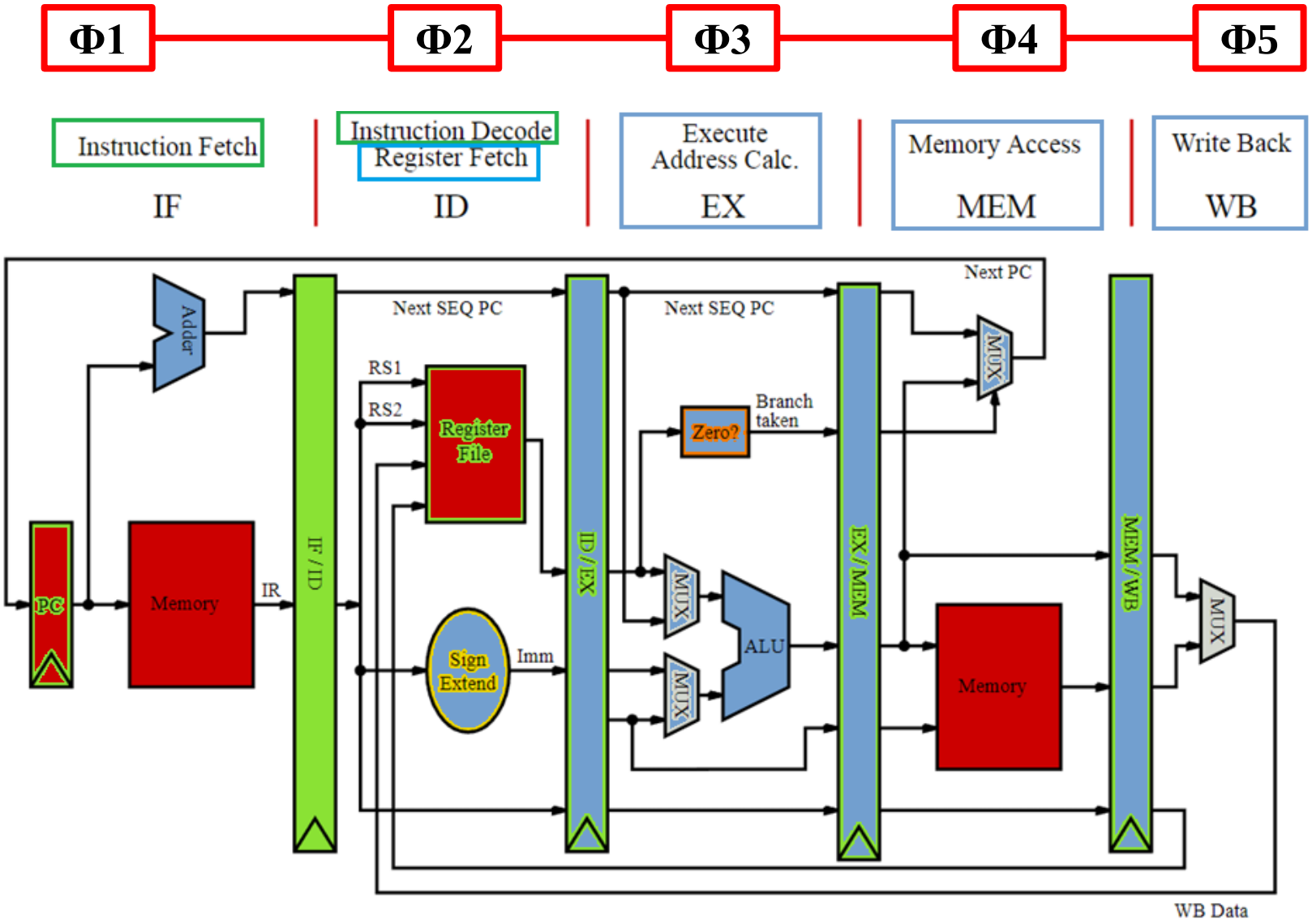


конвейер

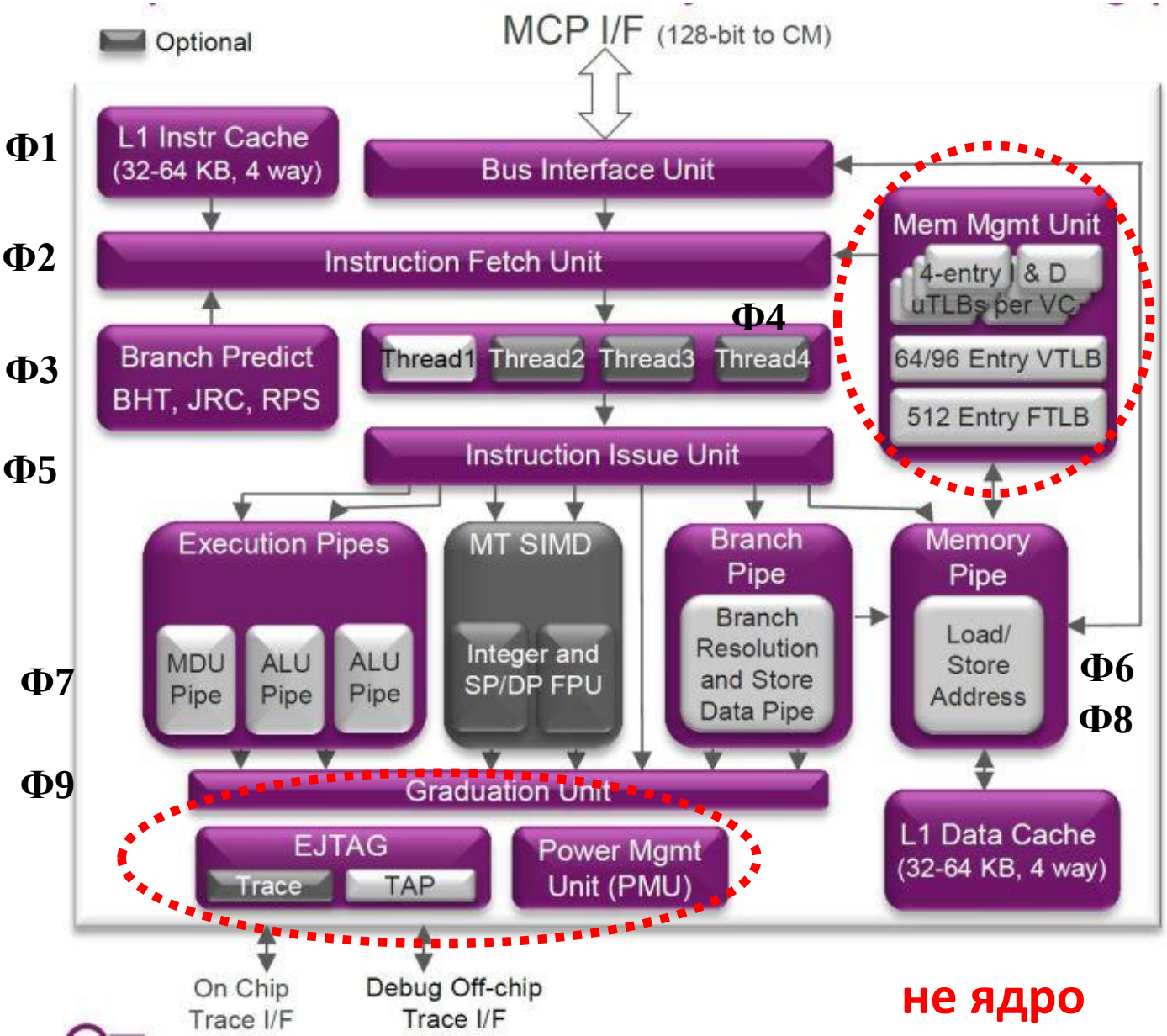
Ф1
Ф2
Ф3
Ф4
...
Ф11
Ф12



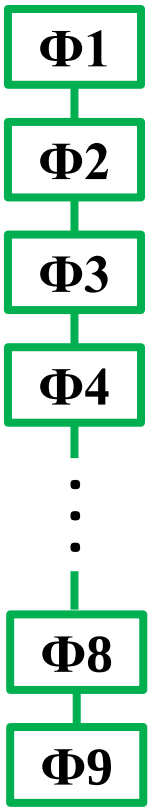
Однотактный пятиступенчатый процессор MIPS



Конвейерная архитектура MIPS

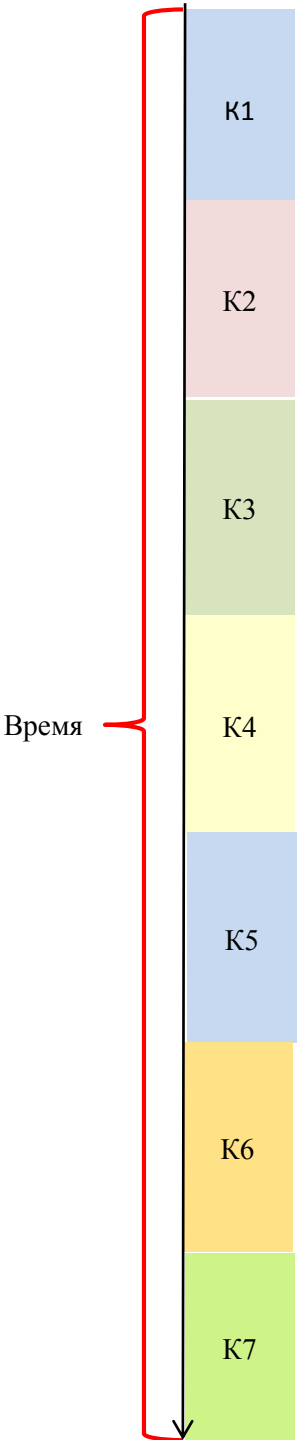


конвейер

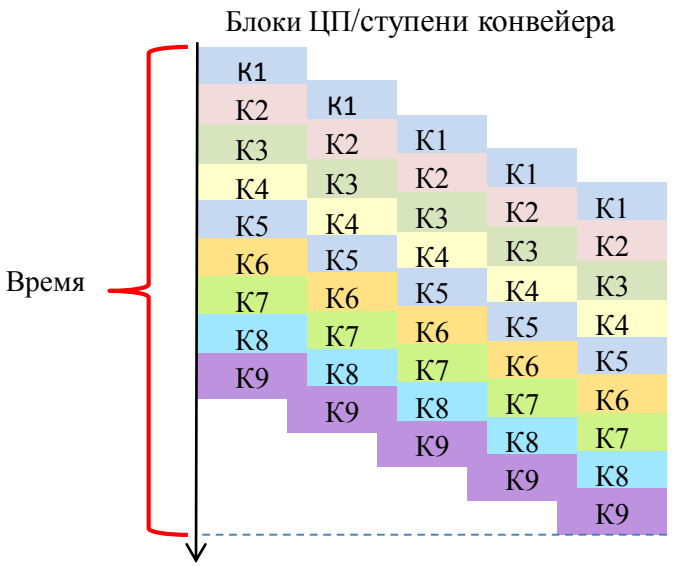


не ядро

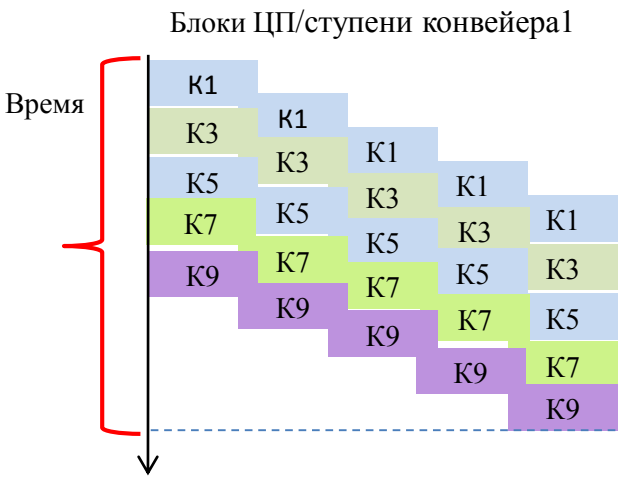
Последовательная обработка

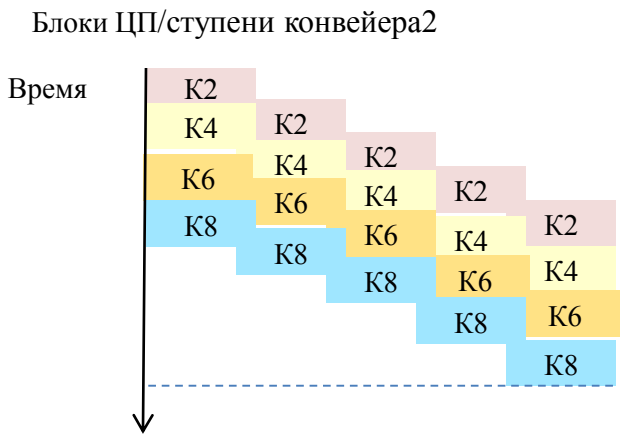


Конвейерная обработка



Несколько конвейеров/потоков





НО

Реально конвейер увеличивает скорость выполнения программы не в 12-24 раз, а гораздо меньше.

- Микроархитектура процессора
- Структура программы

} *скорость*

Простой, вызванный взаимозависимостью команд по данным

```
SUB     EDX, EDX
MOV     AX, MEM1
MOVZX   EAX, AX
DIV     MEM2
CMP     EDX,0
JZ      ZERO
PUSH    STR1
```

Пока команда **MOV AX, MEM1** не закончит запись числа из памяти в регистр AX, следующая команда **MOVZX EAX, AX** не сможет завершить выборку операндов и выполнить операцию расширения.

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
11	JZ ZERO	CMP EDX,0	DIV MEM2		
12	JZ ZERO	CMP EDX,0	DIV MEM2		
13	JZ ZERO	CMP EDX,0	DIV MEM2		
14	JZ ZERO	CMP EDX,0	DIV MEM2		
…	JZ ZERO	CMP EDX,0	(КЭШ-промах) Поиск в ОП		
25		JZ ZERO	CMP EDX,0	DIV MEM2	
26		JZ ZERO	CMP EDX,0		DIV MEM2
27		JZ ZERO	CMP EDX,0		DIV MEM2
28		JZ ZERO	CMP EDX,0		
29			JZ ZERO	CMP EDX,0	
30			JZ ZERO		CMP EDX,0
31			JZ ZERO		
32				JZ ZERO	
33					JZ ZERO
34	PUSH STR1				
35		PUSH STR1			
36			PUSH STR1		
37				PUSH STR1	
38					PUSH STR1

Для CMP на готов остаток от деления – регистр EDX, для JZ не готов флаг ZF

Простой, вызванный различной длительностью разных фаз команд (следующая ступень конвейера занята)

Фаза 3 команды **MOV AX, MEM1** и фаза 1 команды **DIV MEM2** и фаза 2 команды **MOVZX EAX, AX** начинаются одновременно на такте 5. Но в первом случае длительность фазы составит 3 такта, а во втором и третьем случае – 1 такт. Значит не смотря на то, что вторая и третья команда готова перейти в следующий блок конвейера, она этого сделать не может, т.к. этот блок занят предыдущей командой. И в течении следующих 3 тактов первая и вторая команда (первая и вторая ступень конвейера) простаивает.

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
11	JZ ZERO	CMP EDX,0	DIV MEM2		
12	JZ ZERO	CMP EDX,0	DIV MEM2		
13	JZ ZERO	CMP EDX,0	DIV MEM2		
14	JZ ZERO	CMP EDX,0	DIV MEM2		
…	JZ ZERO	CMP EDX,0	(КЭШ-промах) Поиск в ОП		
25		JZ ZERO	CMP EDX,0	DIV MEM2	
26		JZ ZERO	CMP EDX,0		DIV MEM2
27		JZ ZERO	CMP EDX,0		DIV MEM2
28		JZ ZERO	CMP EDX,0		
29			JZ ZERO	CMP EDX,0	
30			JZ ZERO		CMP EDX,0
31			JZ ZERO		
32				JZ ZERO	
33					JZ ZERO
34	PUSH STR1				
35		PUSH STR1			
36			PUSH STR1		
37				PUSH STR1	
38					PUSH STR1

Простой, вызванный различной длительностью разных фаз команд (команда не готова перейти с предыдущей ступени)

Например

Ступень 4 конвейера (обрабатывающие блоки) простаивает с 5 по 25 такт, т.к. не подготовлены данные ни по одной команде обработки данных.

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
11	JZ ZERO	CMP EDX,0	DIV MEM2		
12	JZ ZERO	CMP EDX,0	DIV MEM2		
13	JZ ZERO	CMP EDX,0	DIV MEM2		
14	JZ ZERO	CMP EDX,0	DIV MEM2		
…	JZ ZERO	CMP EDX,0	(КЭШ-промах) Поиск в ОП		
25		JZ ZERO	CMP EDX,0	DIV MEM2	
26		JZ ZERO	CMP EDX,0		DIV MEM2
27		JZ ZERO	CMP EDX,0		DIV MEM2
28		JZ ZERO	CMP EDX,0		
29			JZ ZERO	CMP EDX,0	
30			JZ ZERO		CMP EDX,0
31			JZ ZERO		
32				JZ ZERO	
33					JZ ZERO
34	PUSH STR1				
35		PUSH STR1			
36			PUSH STR1		
37				PUSH STR1	
38					PUSH STR1

Простой, вызванный КЭШ промахом и длительным поиском данных в оперативной памяти ВС

Простои возникают при обращении за данными в ОП. Но процессор не обращается в ОП напрямую, а ищет нужные данные в КЭШ, и если не находит, то только тогда выполняет длительную операцию чтения блока ОП. Простои возникают случайно и непредсказуемо, и тормозят блоки как до выборки операндов, так и после.

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
11	JZ ZERO	CMP EDX,0	DIV MEM2		
12	JZ ZERO	CMP EDX,0	DIV MEM2		
13	JZ ZERO	CMP EDX,0	DIV MEM2		
14	JZ ZERO	CMP EDX,0	DIV MEM2		
…	JZ ZERO	CMP EDX,0	(КЭШ-промах) Поиск в ОП		
25		JZ ZERO	CMP EDX,0	DIV MEM2	
26		JZ ZERO	CMP EDX,0		DIV MEM2
27		JZ ZERO	CMP EDX,0		DIV MEM2
28		JZ ZERO	CMP EDX,0		
29			JZ ZERO	CMP EDX,0	
30			JZ ZERO		CMP EDX,0
31			JZ ZERO		
32				JZ ZERO	
33					JZ ZERO

Простои в работе конвейера команд

SUB EDX, EDX
MOV AX, MEM1
MOVZX EAX, AX
DIV MEM2
CMP EDX,0
JZ ZERO
PUSH STR1

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
11	JZ ZERO	CMP EDX,0	DIV MEM2		
12	JZ ZERO	CMP EDX,0	DIV MEM2		
13	JZ ZERO	CMP EDX,0	DIV MEM2		
14	JZ ZERO	CMP EDX,0	DIV MEM2		
…	JZ ZERO	CMP EDX,0	(КЭШ-промах) Поиск в ОП		
25		JZ ZERO	CMP EDX,0	DIV MEM2	
26		JZ ZERO	CMP EDX,0		DIV MEM2
27		JZ ZERO	CMP EDX,0		DIV MEM2
28		JZ ZERO	CMP EDX,0		
29			JZ ZERO	CMP EDX,0	
30			JZ ZERO		CMP EDX,0
31			JZ ZERO		
32				JZ ZERO	
33					JZ ZERO
34	PUSH STR1				
35		PUSH STR1			
36			PUSH STR1		
37				PUSH STR1	
38					PUSH STR1

Простой, вызванный различной длительностью разных фаз команд (следующая ступень конвейера занята)

Простой из-за КЭШ-промаха

Простой, вызванный различной длительностью разных фаз команд (команда не готова перейти с предыдущей ступени)

Простой из-за зависимости по данным (регистр AX, EDX, регистр флагов–бит ZF

Простой после команды передачи управления

ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

- *Простой, вызванный различной длительностью разных фаз команд*
- *Простой после команды передачи управления*
- *Простой из-за зависимости команд по данным*
- *Простой, вызванный КЭШ-промахом*

Предложите способы борьбы с этими простоями

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО РАЗНОЙ ДЛИТЕЛЬНОСТЬЮ РАЗНЫХ ФАЗ КОМАНДЫ

простой ступени1
на тактах 3-4

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К2	К1		
4	К2	К1		
5		К2	К1	
6			К2	К1
7				К2

простой ступеней
3,4 на тактах 4-6

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3		К2	К1	
4		К2		К1
5		К2		
6			К2	
7				К2

? ? ? ? ? ?

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО РАЗНОЙ ДЛИТЕЛЬНОСТЬЮ РАЗНЫХ ФАЗ КОМАНДЫ

простой ступени1
на тактах 3-4

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К2	К1		
4	К2	К1		
5		К2	К1	
6			К2	К1
7				К2

простой ступеней
3,4 на тактах 4-6

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3		К2	К1	
4		К2		К1
5		К2		
6			К2	
7				К2

- уравнивание длительности всех фаз,

УРАВНИВАНИЕ ДЛИТЕЛЬНОСТИ ВСЕХ ФАЗ (до самой длинной)

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К2	К1		
4	К2	К1		
5		К2	К1	
6			К2	К1
7				К2

длительность фазы = длительность цикла = $\frac{1}{\text{тактовая частота}}$

- Упрощает управление
- Маскирует простои

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К1			
3	К1			
4	К2	К1		
5	К2	К1		
6	К2	К1		
7		К2	К1	
8		К2	К1	
9		К2	К1	
10			К2	К1
11			К2	К1

ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

Как ликвидировать простой, вызванный различной длительностью разных фаз команд?

Условно две категории простоев:

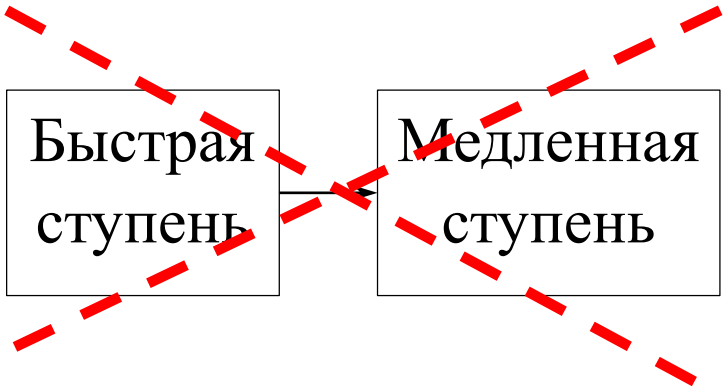
1)Блок занят

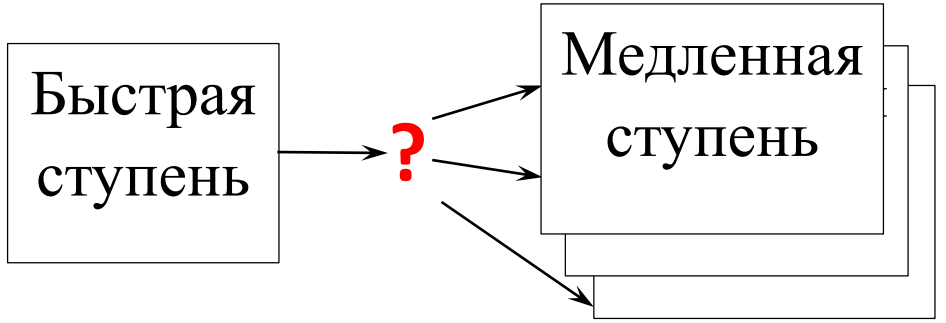
***Продублировать блоки**, занимающие много времени на исполнение фазы*

2)Фаза команды исполняется слишком долго

*Одна из фаз команды слишком длинная – надо её **разбить на несколько** более мелких.*

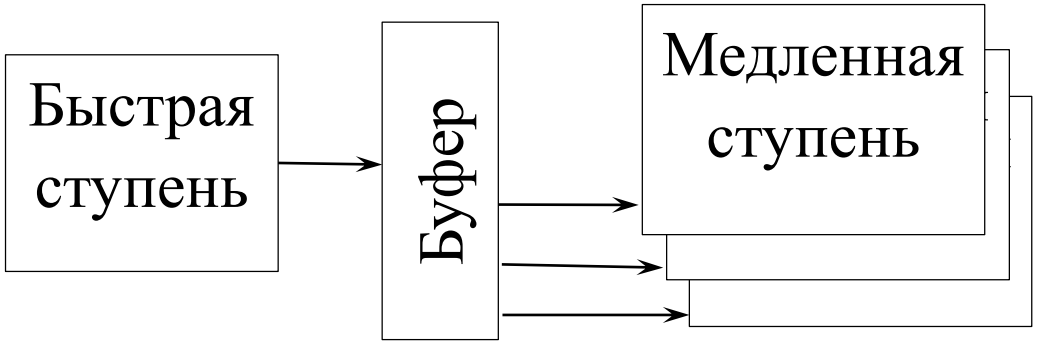
Дублировать медленные или наиболее востребованные ступени (блоки ЦП)





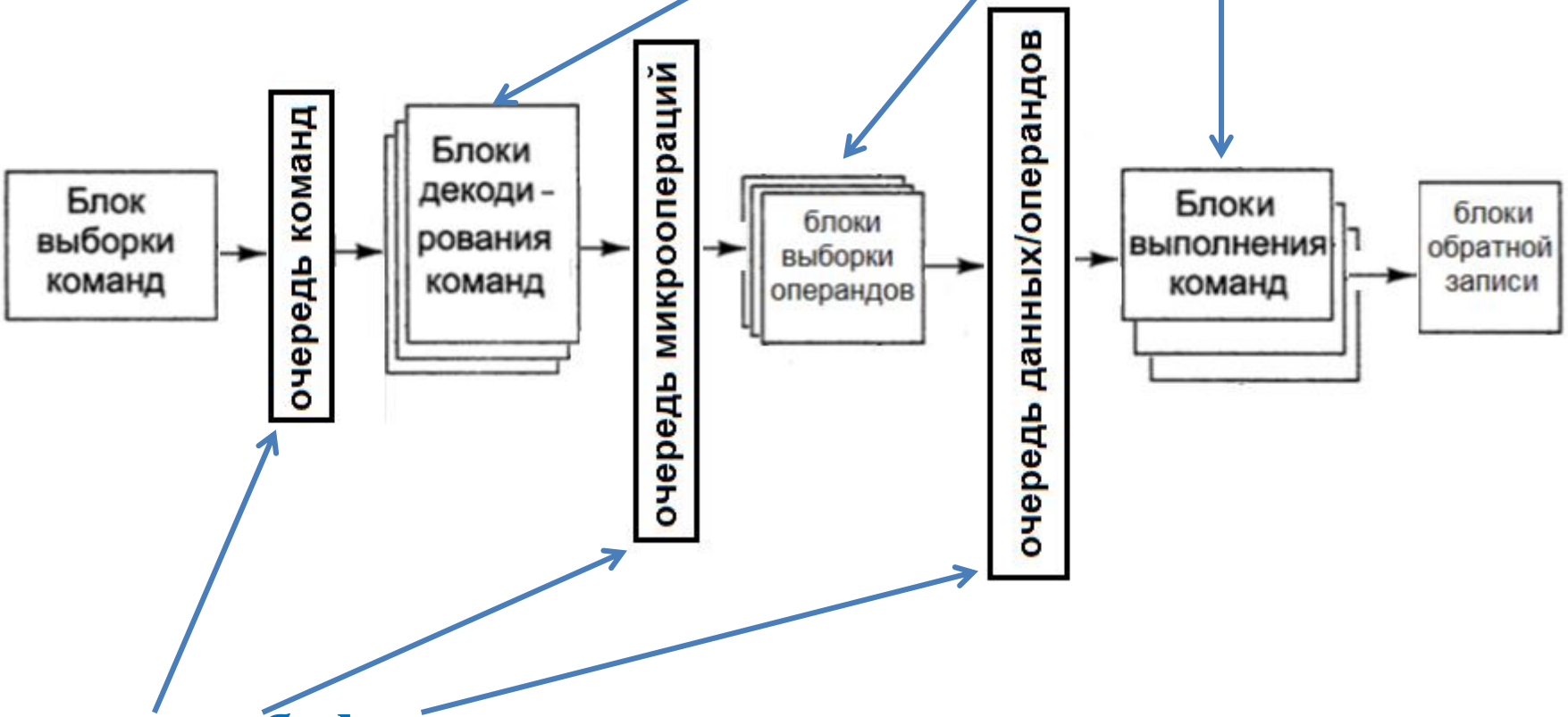
такты	Ступень1	Ступень2.1	Ступень2.2	Ступень2.3	Ступень3	Ступень4
1.	К1					
1.	К2	К1				
1.	К3	К1	К2			
1.		К1	К2	К3		
1.			К2	К3	К1	
1.				К3	К2	К1
1.					К3	К2
1.						К3

ВЕДЕНИЕ БУФЕРОВ между ступенями с разной длительностью фаз (блоками ЦП)



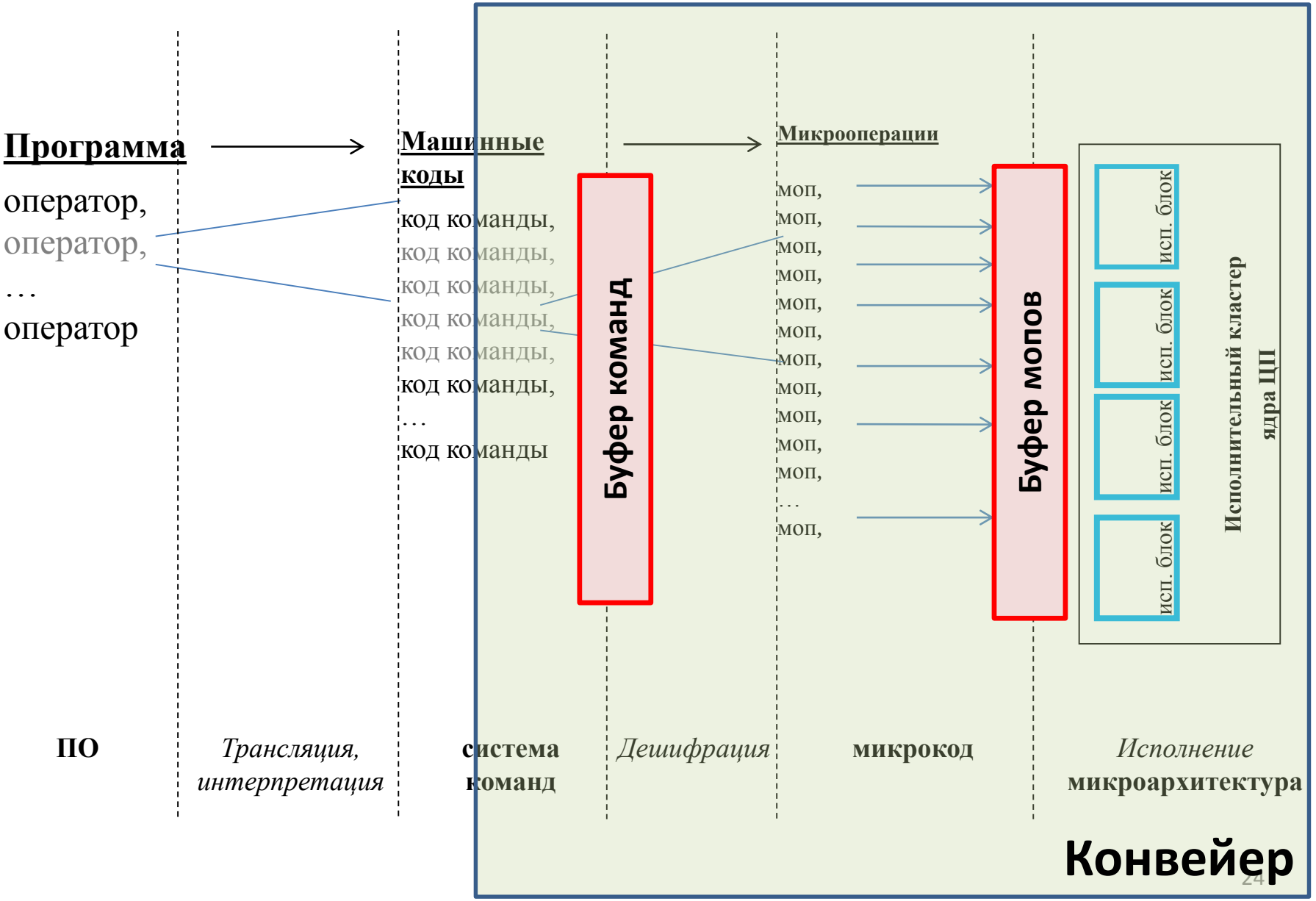
Способ ликвидации простоев, вызванных различной длительностью разных фаз команд

если блок часто занят – **дублировать блоки**

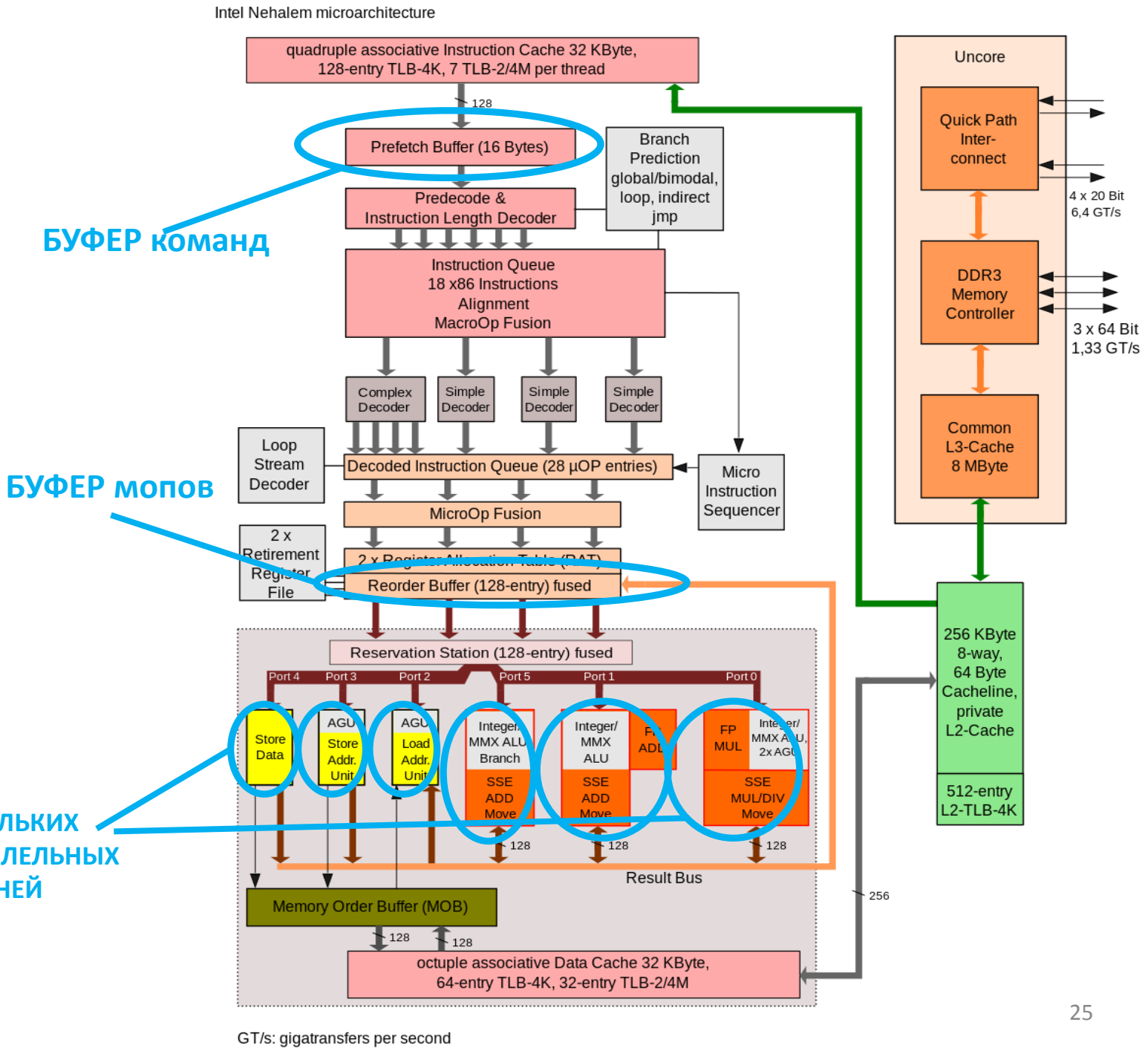


и **вводить буферы** между ступенями конвейера (блоками ЦП) для распределения команд/мопов по нескольким блокам)

ВЕДЕНИЕ БУФЕРОВ МЕЖДУ СТУПЕНЯМИ С РАЗНОЙ ДЛИТЕЛЬНОСТЬЮ ФАЗ И ДОБАВЛЕНИЕ НЕСКОЛЬКИХ ПАРАЛЛЕЛЬНЫХ МЕДЛЕННЫХ СТУПЕНЕЙ



Ядро ЦП с микроархитектурой Intel Nehalem

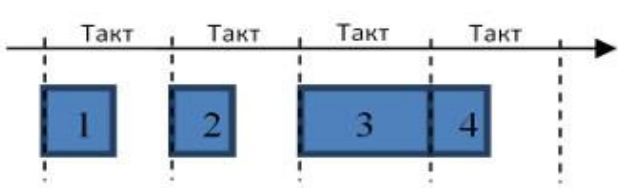


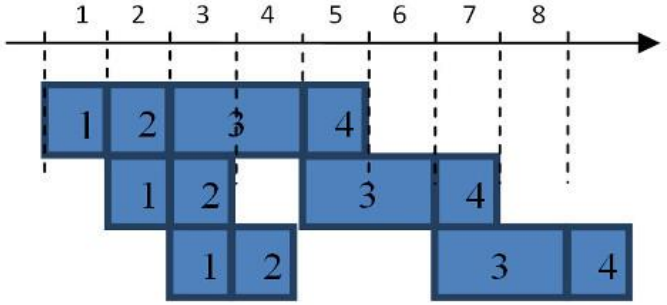
ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

Как ликвидировать простой, вызванный различной длительностью разных фаз команд ?

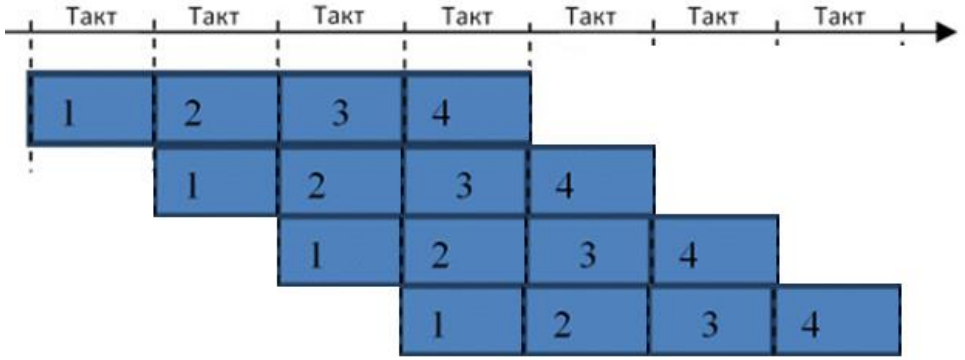
2) Если одна из фаз команды слишком длинная – надо её **разбить на несколько более мелких**. Когда разбиение более не возможно, уравнять все фазы – по длине самой долгой.

Ликвидации простоев (неравномерность фаз).

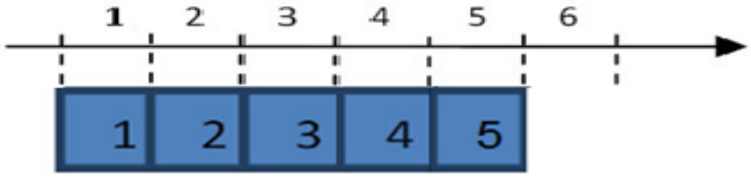


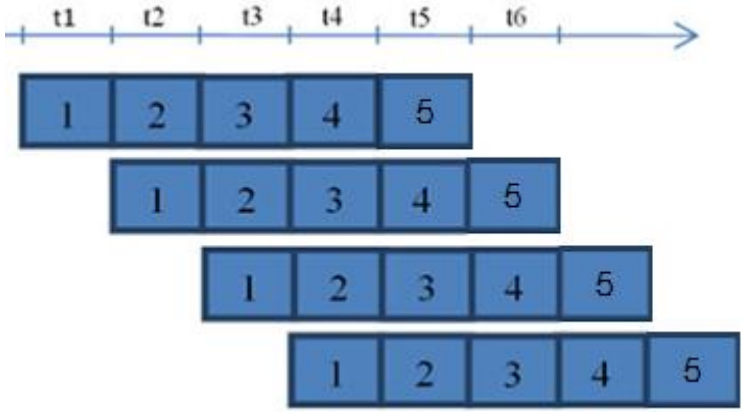


Первый способ (для простоты управления)– уровнять фазы и сделать длительность такта по длительности максимальной фазы



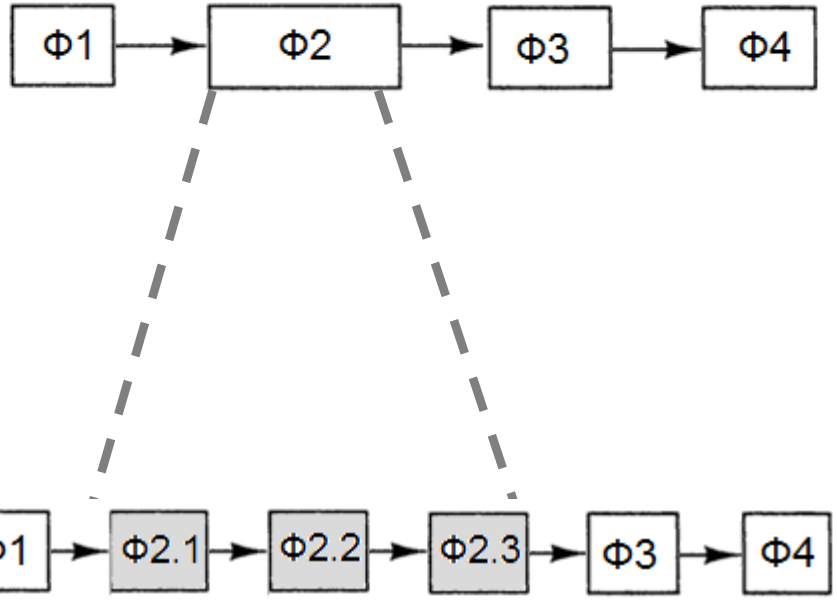
Второй способ – сократить длительность такта, порезать самые длинные фазы на более мелкие операции и создать отдельные блоки для их исполнения





РАЗБИЕНИЕ ДЛИННЫХ ФАЗ НА НЕСКОЛЬКО КОРОТКИХ

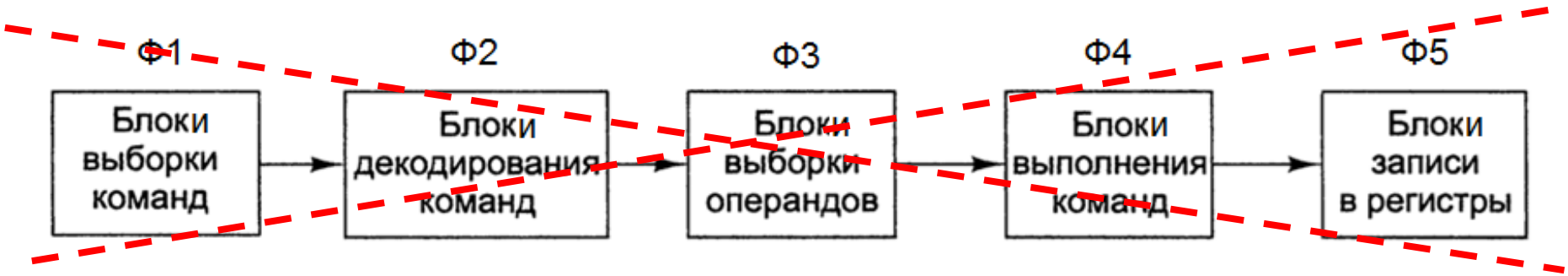
такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К2	К1		
4	К2	К1		
5		К2	К1	
6			К2	К1
7				К2



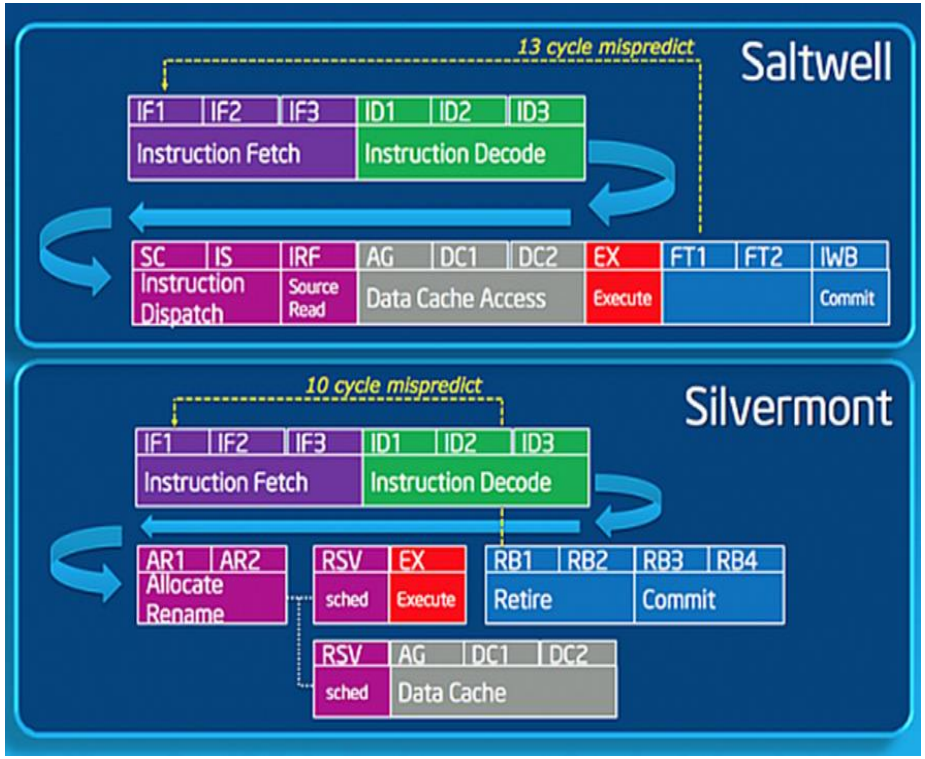
такты	Ступень1	Ступень2.1	Ступень2.2	Ступень2.3	Ступень3	Ступень4
1	К1					
2	К2	К1				
3		К2	К1			
4			К2	К1		
5				К2	К1	
6					К2	К1
7						К2

разбиение одного сложного блока ЦП на несколько более простых

РАЗБИЕНИЕ ДЛИННЫХ ФАЗ НА НЕСКОЛЬКО КОРОТКИХ



пятиступенчатый конвейер Intel 486 и Pentium

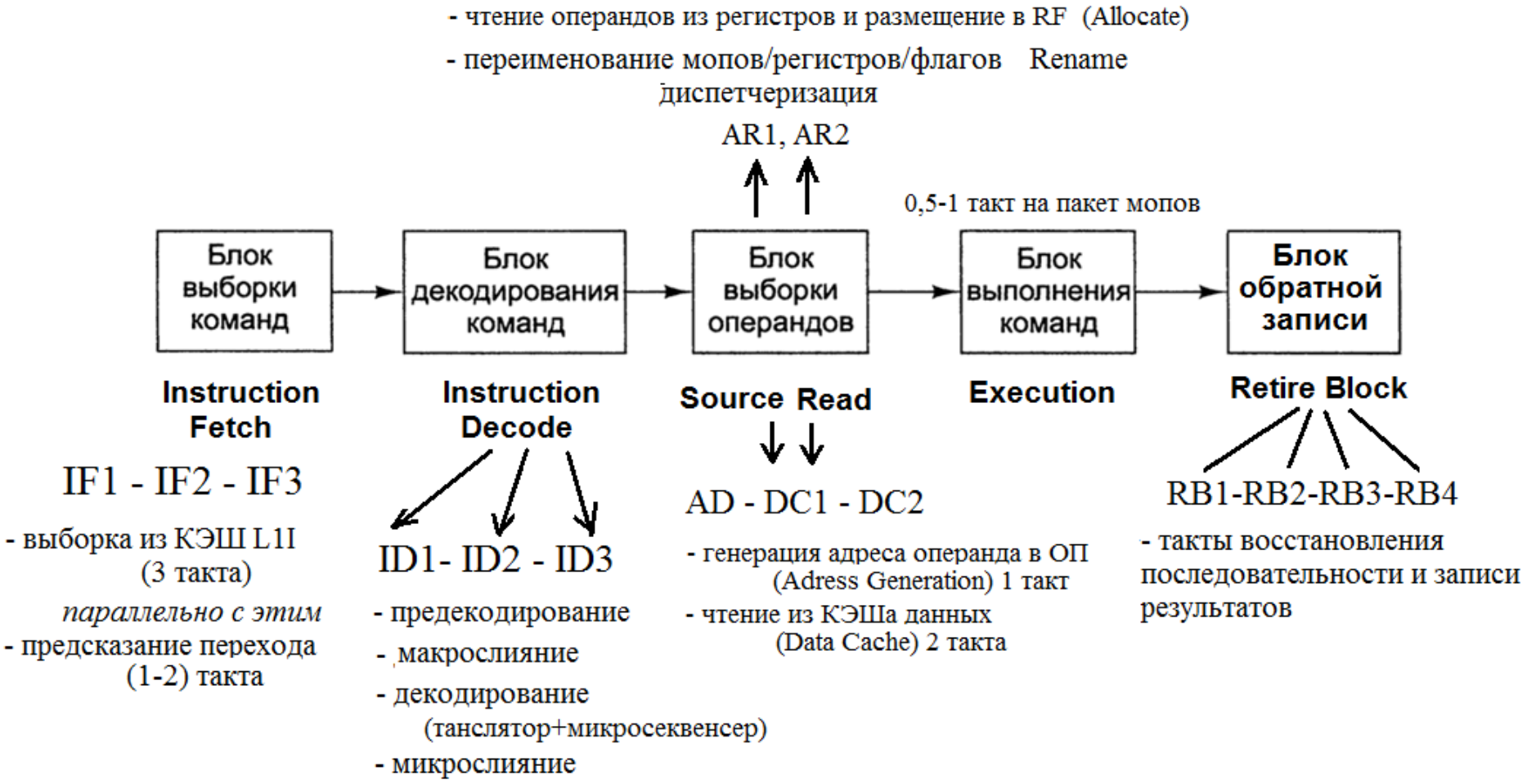


Фазы		
раньше	new	название
Ф1. IF – выборка команды	1	IF1
	2	IF2
	3	IF3
Ф2. ID - декодирование	4	ID1
	5	ID2
	6	ID3
Изменение последовательности команд	7	AR1/SC
	8	AR2/IS
	9	IRF
Ф3. Выборка операндов	10	AG
	11	DC1
	12	DC2
Ф4. Исполнение операции	13	EX1
Ф5. Обратная запись	14	FT1
	15	FT2
	16	IWB/DC

16 (14/12)-ступенчатый конвейер Intel Saltwell/Atom и Intel Silvermont/Ivy Bridge (2012-2013)

Ликвидации простоев 3 и 4 типа (неравномерность фаз) – разбиение длинных фаз на несколько более коротких

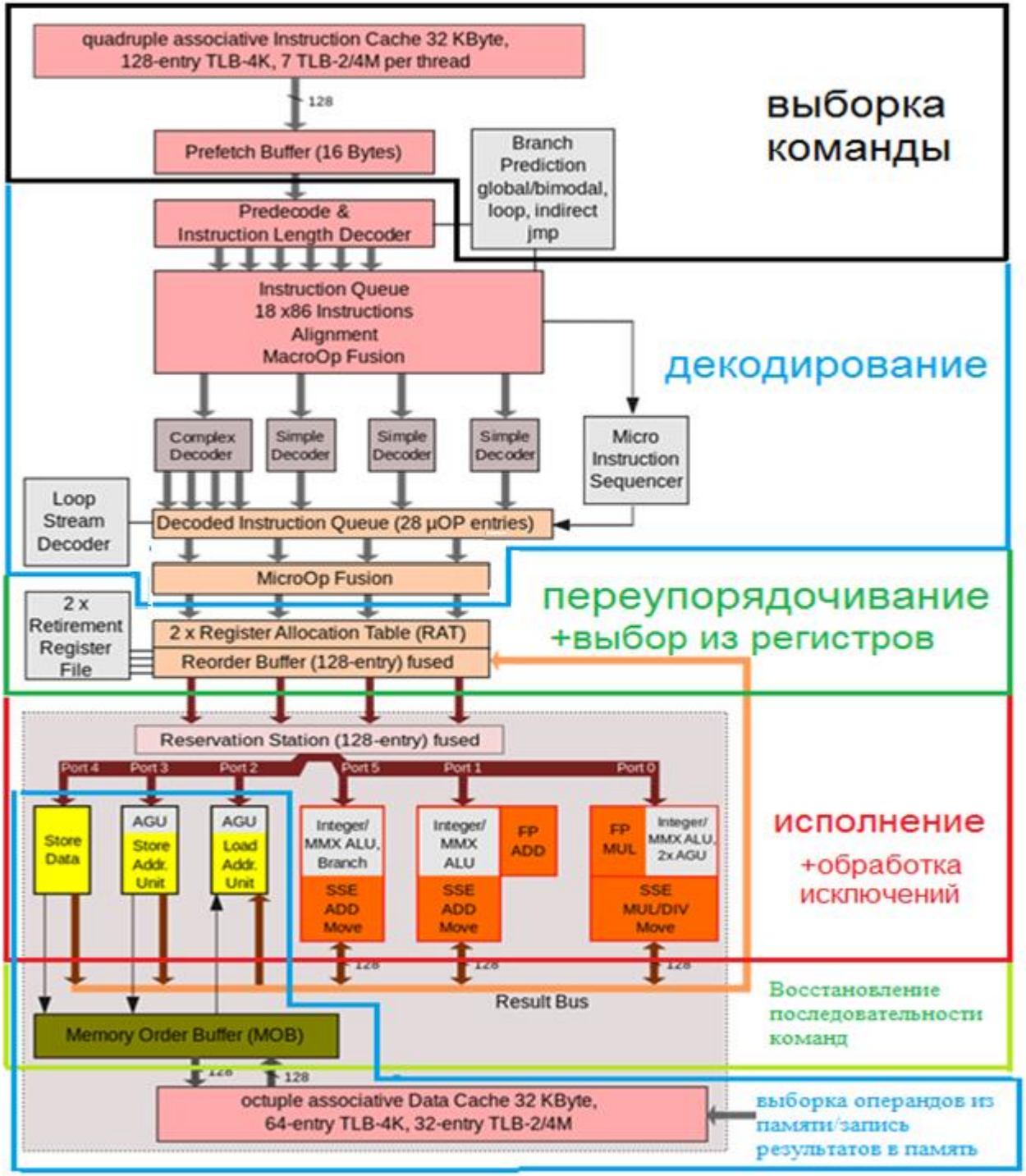
Современные процессоры имеют порядка двух десятков стадий в конвейере



http://www.ixbt.com/cpu/cpu-pedia.shtml **Энциклопедия процессорных терминов**

Современные процессоры с разбиением команды на 16 фаз/стадий

Стадии		Выполняемые действия		
1	IF1	Выборка из КЭШ L1I в блока предварительной выборки	BPU	Предсказание адреса перехода
2	IF2			
3	IF3		ILD	Перекодирование - определения длин команд во фрагменте кода
4	ID1	Декодирование		
5	ID2	Макрослияние - позволяет одним мопом закодировать две (редко больше) команды		
6	ID3	Микрослияние - позволяет одним мопом закодировать две микрооперации		
7	AR1	Размещение – чтение и размещение содержимого архитектурных регистров в регистровом файле и буфере мопов		
8	AR2	Переименование операндов/флагов в соответствии с размещением в регистровом файле (заполнение таблицы RAT)		
9	RSV	Диспетчеризация - планирование и осуществление внеочередного запуска на исполнение мопов		
10	EX1	Исполнение (Execution) параллельное исполнение нескольких слитных мопов в нескольких исполнительных блоках	AG	Генерация адреса для операнда в ОП
11	EX1		DC1	Доступ к L1D (Data cache) поиск операнда
12	EX1		DC2	Выборка операнда и передача в буфер мопов
13	RB1	Перенесение результатов исполненных мопов в регистровый файл		
14	RB2	Корректировка используемого для переименования регистров таблицы ссылок на физический РФ, чтобы записанный мопом архитектурный регистр указывал на верный физический		
15	RB3	Проверка на очередность записи результатов и на возможность завершения команды. Отказ возможен в случае обнаружения: • исключения при исполнении мопа; • для условных переходов — неверного предсказания перехода (поведения или адреса); • для мопов, выполнивших упреждающие чтения из памяти — неверного предсказания адреса. В последних двух случаях диспетчер возвращает конвейер в предыдущее точно известное состояние («сброс конвейера»), теряя все упреждающие результаты.		
16	RB4	Обратная запись результатов, готовых на текущий так в правильном порядке		



1) Формирование адреса команды
2)Поиск в КЭШ L1I
3)Передача широкого слова в блок предварительной выборки

1)Предекодерование (определение длины /сложности команд, выбор декодера или передача в планировщик (MIS),
2)Макрослияние
3)Декодирование = формирование мопов
4)Микрослияние

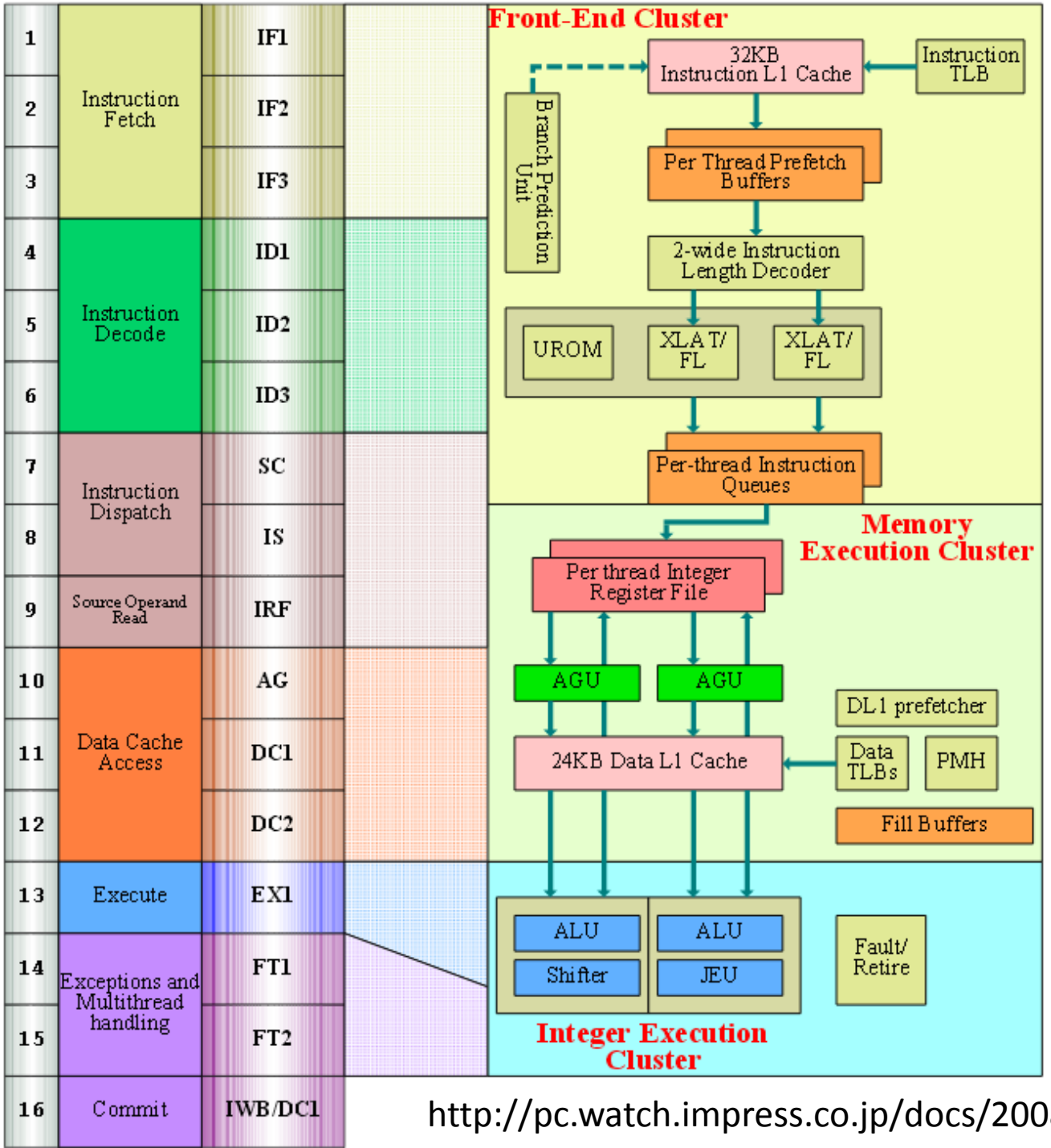
3)Переупорядочивание мопов, переименование регистров
2)Анализ типов мопов (int/float/sse…), слияние совместимых и распределение слитых по исполнительным кластерам –
3)Загрузка операнда из связанного регистра в ROB

1)Генерация адреса операнда в ОП
2)доступ к L1D (Data cache) поиск операнда
3)загрузка операнда из L1D в буфер конвейер операций

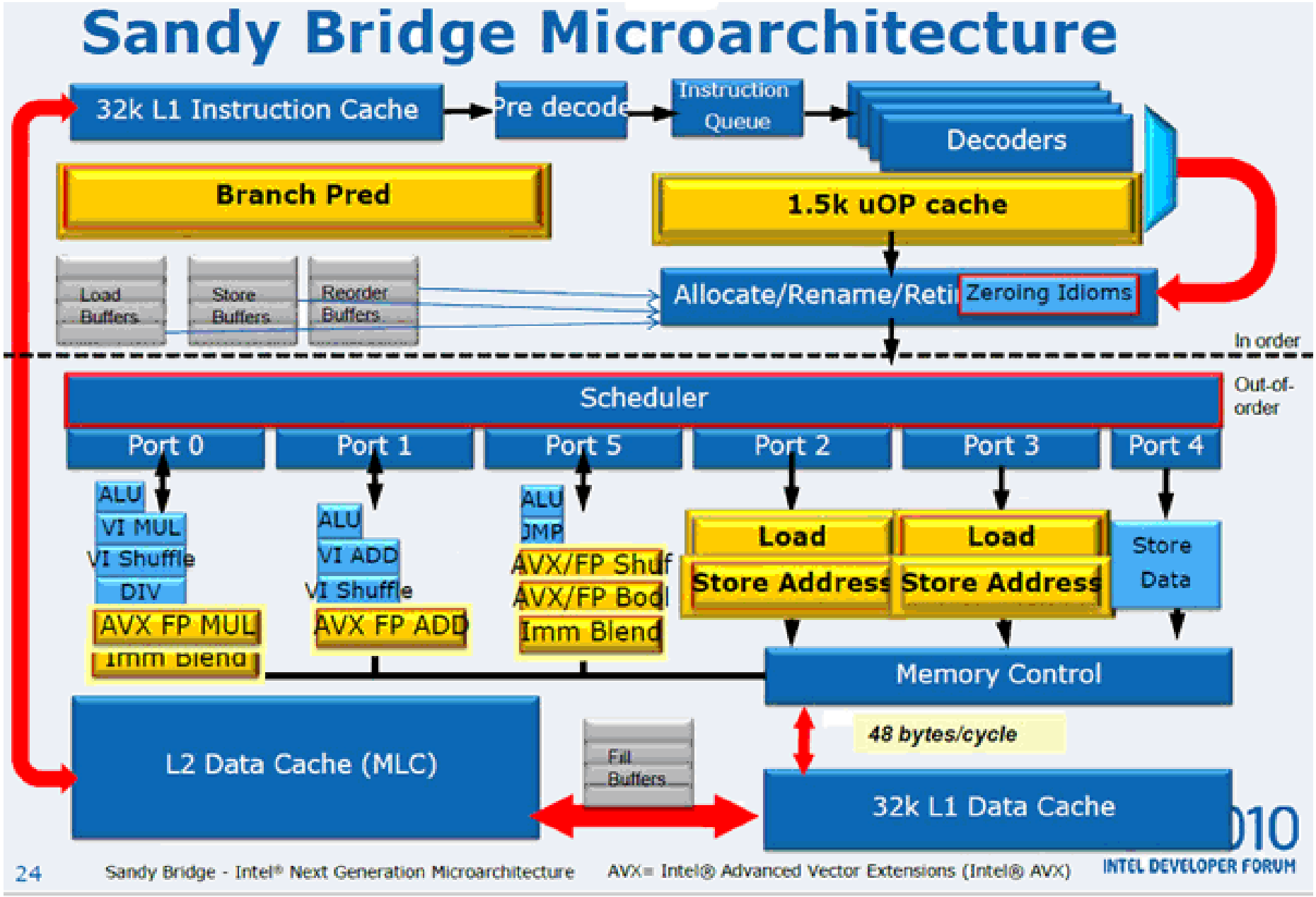
1) параллельное исполнение нескольких моп в нескольких исполнительных блоках и запись результатов в RB
2)Проверка исключений

1)Сканирование RB на предмет обнаружения мопов, которые уже не повлияют на выполнение других команд и формирование из них совокупности завершённых команд (для простых инструкций)
Восстановление последовательности результатов моп в завершённых командах в порядке поступления команд
Обратная запись результатов

Конвейер Silverthorne



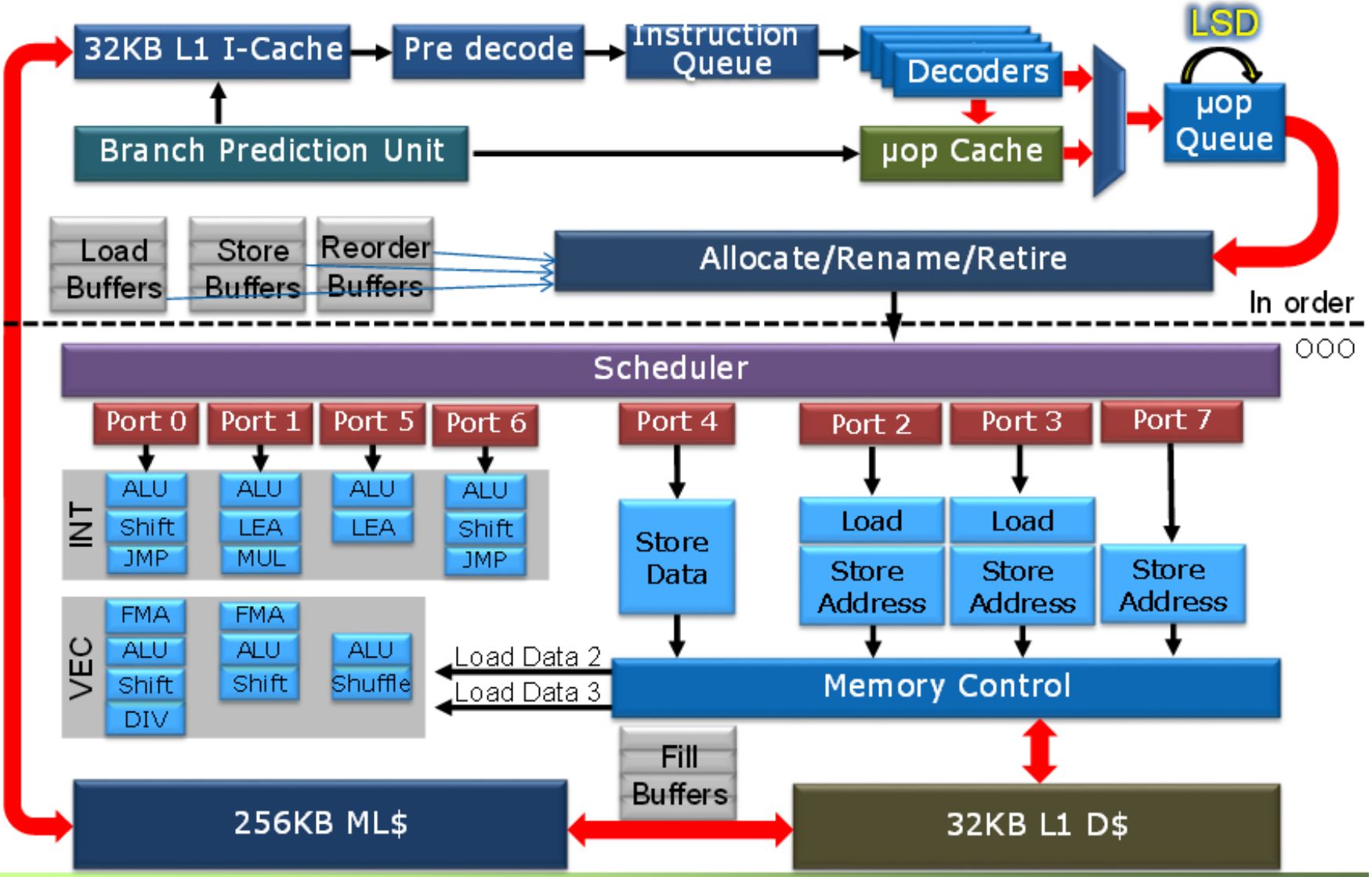
http://pc.watch.impress.co.jp/docs/2008/0207/kaigai417_05.pdf



http://hardwareguide.ru/%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80/sandy-bridge/6/

http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf

Haswell Core µArch



ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

- *Простой, вызванный различной длительностью разных фаз команд* √
- *Простой из-за зависимости команд по данным*
- *Простой после команды передачи управления*
- *Простой, вызванный КЭШ-промахом*

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО ЗАВИСИМОСТЬЮ КОМАНД ПО ДАННЫМ

Команда К2 использует результаты команды К1, которые будут готовы на 4 такте. Значит К2 может их использовать не ранее 5 такта.

простой ступеней 2,3,4 на тактах 3-6

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К3		К1	
4				К1
5		К2		
6		К3	К2	
7			К3	К2

Предложите способы борьбы с этими простоями

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО ЗАВИСИМОСТЬЮ КОМАНД ПО ДАННЫМ

Команда К2 использует результаты команды К1, которые будут готовы на 4 такте. Значит К2 может их использовать не ранее 5 такта.

простой ступеней 2,3,4 на тактах 3-6

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К2	К1		
3	К3		К1	
4				К1
5		К2		
6		К3	К2	
7			К3	К2

поможет

- Внеочередное исполнение команд

ВНЕОЧЕРЕДНОЕ ИСПОЛНЕНИЕ КОМАНД

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2	К3	К1		
3	К4	К3	К1	
4	К2	К4	К3	К1
5		К2	К4	К3
6			К2	К4
7				К2

Команда К2 может запуститься на ступени 2 не ранее 5 такта, тогда до неё на тактах 2,3,4 будут запущены более поздние команды К3 и К4 (при условии, что они по этим же данным независимы).

Очень сильно усложняется структура ЦП:

- дополнительные накопительные буферы отложенных команд/микроопераций
- дополнительные накопительные буферы для хранения готовых операндов
- дополнительные блоки управления и диспетчеризации потока команд/ микроопераций
- дополнительные блоки восстановления правильного порядка потока результатов команд/микроопераций

Ликвидация простоев, вызванных зависимостью команд по данным - ИЗМЕНЕНИЕ ОЧЕРЕДНОСТИ ИСПОЛНЕНИЯ КОМАНД

```
SUB    EDX, EDX
MOV    AX, MEM1
MOVZX  EAX, AX
DIV    MEM2
```

Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX	2 такта	MOV AX, MEM1
9	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
10	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX

Простой есть – выборка операндов для MOVZX EAX, AX выполняется за 2 такта

Меняем местами команды

```
MOV    AX, MEM1
SUB    EDX, EDX
```

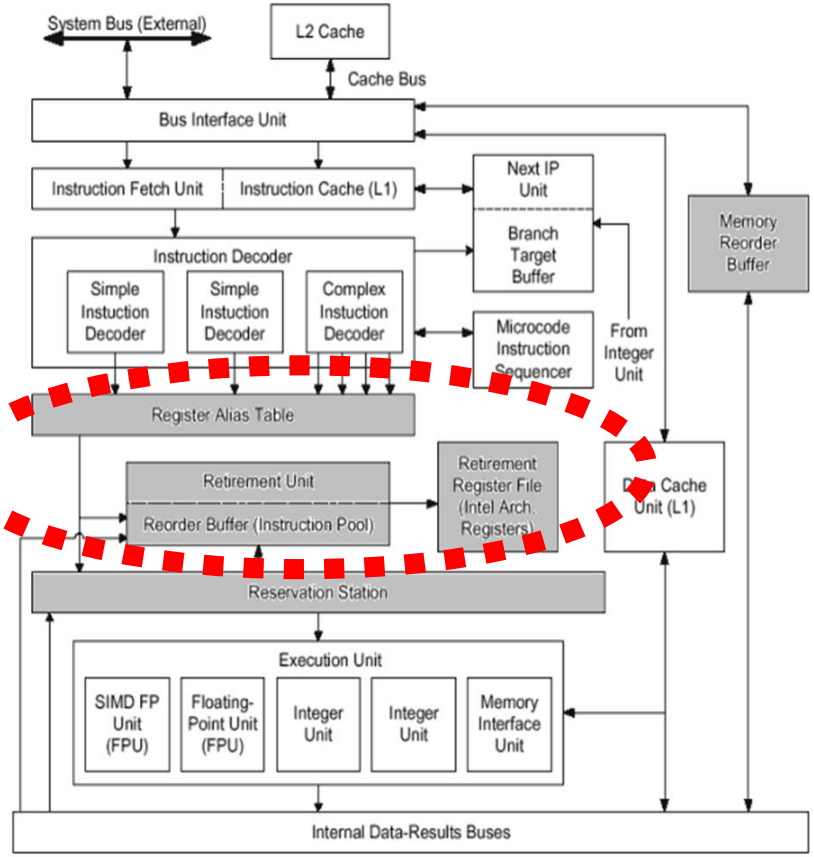
```
MOV    AX, MEM1
SUB    EDX, EDX
MOVZX  EAX, AX
DIV    MEM2
```

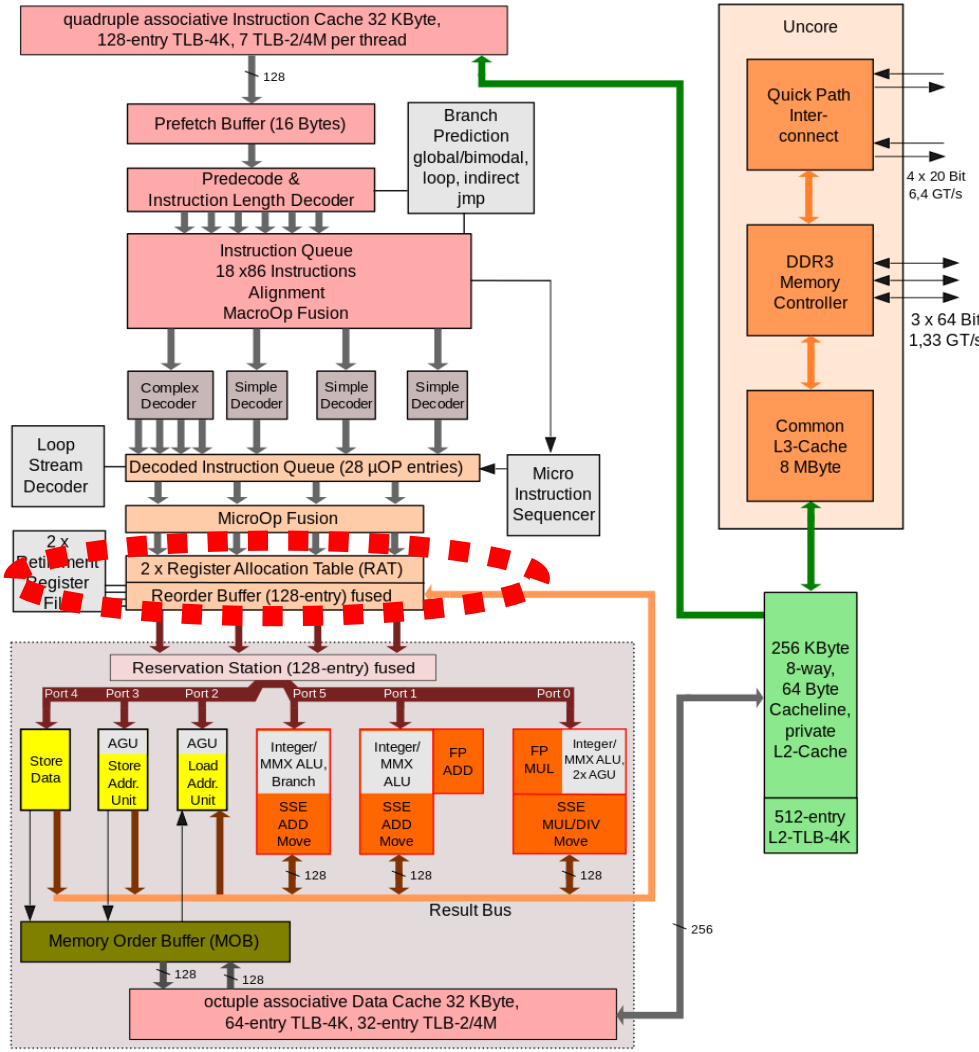
Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	MOV AX, MEM1				
2	SUB EDX, EDX	MOV AX, MEM1			
3	MOVZX EAX, AX	SUB EDX, EDX	MOV AX, MEM1		
4	MOVZX EAX, AX	SUB EDX, EDX	MOV AX, MEM1		
5	MOVZX EAX, AX	SUB EDX, EDX	MOV AX, MEM1		
6	MOVZX EAX, AX	SUB EDX, EDX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	SUB EDX, EDX	1 такт	MOV AX, MEM1
8	CMP EDX,0	DIV MEM2	MOVZX EAX, AX	SUB EDX, EDX	
9	JZ ZERO	CMP EDX,0	DIV MEM2		SUB EDX, EDX
10	JZ ZERO	JZ ZERO	DIV MEM2		MOVZX EAX, AX

Простоя нет – выборка операндов для MOVZX EAX, AX выполняется за 2 такта

Ликвидация простоев второго типа (зависимость по данным)
БЛОКИ ПЕРЕУПОРЯДОЧИВАНИЯ

Нужно размещать зависимые команды (микрооперации) в потоке исполнения по возможности дальше друг от друга, для чего в некоторых современных ЦП/ядрах используются специальные блоки переупорядочивания и обратного восстановления последовательности микроопераций.





ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

- *Простой, вызванный различной длительностью разных фаз команд* √
- *Простой из-за зависимости команд по данным* √
- *Простой после команды передачи управления*
- *Простой, вызванный КЭШ-промахом*

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО КОМАНДАМИ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Команда К2 использует результаты команды К1 (адрес перехода = адрес следующей команды К2), которые будут готовы на 4 такте. Значит К2 может быть выбрана не ранее 5 такта.

простой всех ступеней на тактах 2-7

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2		К1		
3			К1	
4				К1
5	К2			
6		К2		
7			К2	

Предложите способы борьбы с этими простоями

? ? ? ? ?

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО КОМАНДАМИ ПЕРЕДАЧИ УПРАВЛЕНИЯ

Команда К2 использует результаты команды К1 (адрес перехода = адрес следующей команды К2), которые будут готовы на 4 такте. Значит К2 может быть выбрана не ранее 5 такта.

простой всех ступеней на тактах 2-7

такты	Ступень1	Ступень2	Ступень3	Ступень4
1	К1			
2		К1		
3			К1	
4				К1
5	К2			
6		К2		
7			К2	

поможет

- **Предсказание переходов**
- **Спекулятивное (по предположению) исполнение команд**

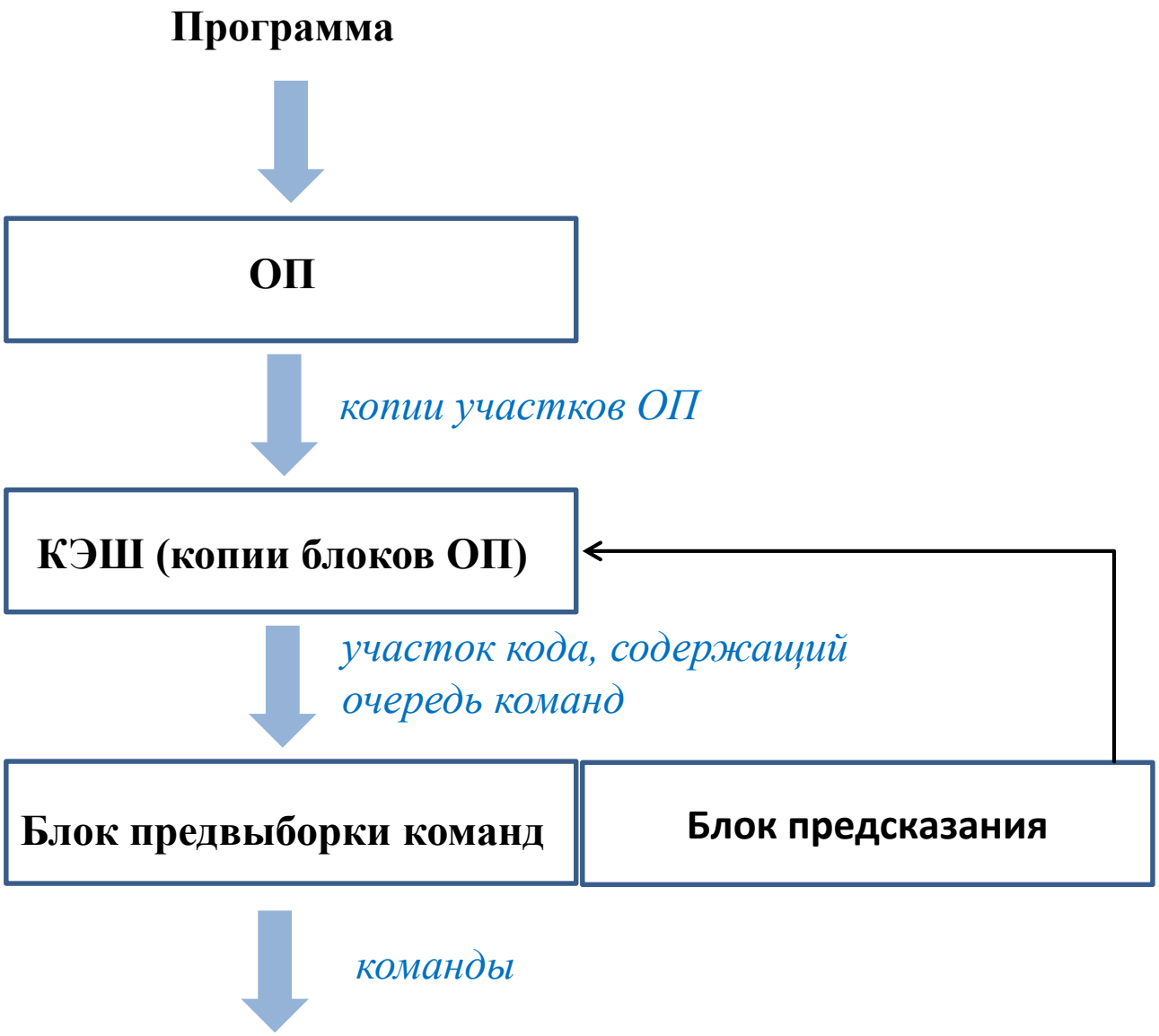
ЗАРАНЕЕ ПРЕДСКАЗЫВАТЬ АДРЕСА СЛЕДУЮЩИХ КОМАНД

Для линейного участка кода программы, состоящего из последовательности команд (обработки /передачи данных и дополнительных) – всё просто. Адрес выбираемого блока команд – следующий за текущим (по порядку).

Для команд передачи управления – сложнее, надо предугадать адрес, куда управление будет передано.

Поведение переходов предсказывается заранее, чаще всего удачно = 93-94% верных предсказаний.

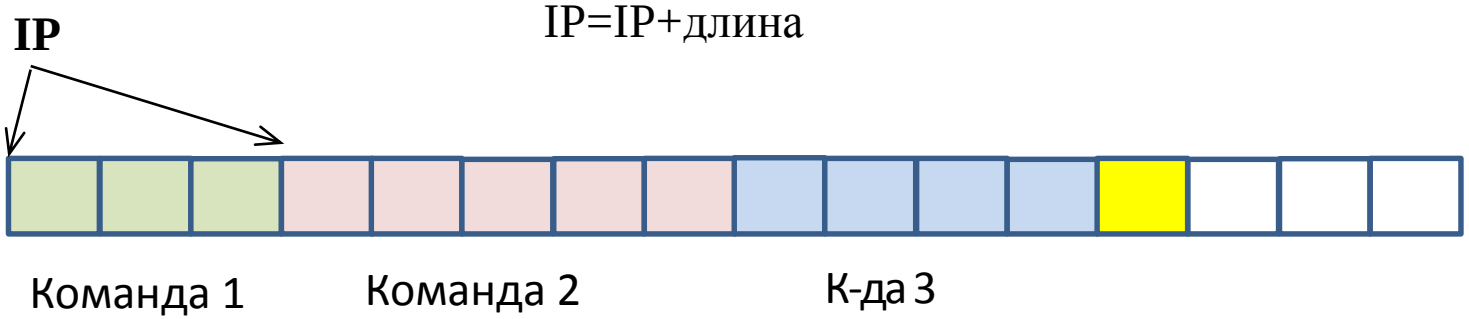
Instruction Cache (L1) *КЭШ инструкций*



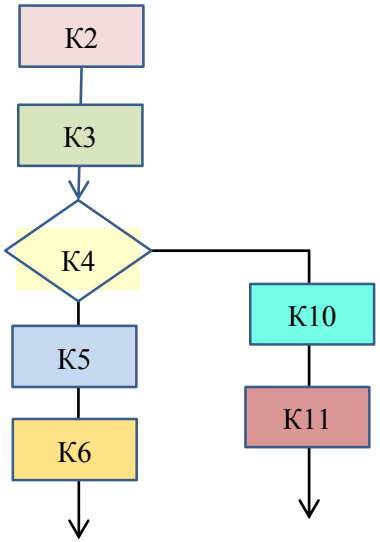
Next IP Unit – блок формирования адреса следующей команды

– рассчитывает адрес следующей команды в потоке исполнения

- Для «линейных» команд



- Для команд ветвления

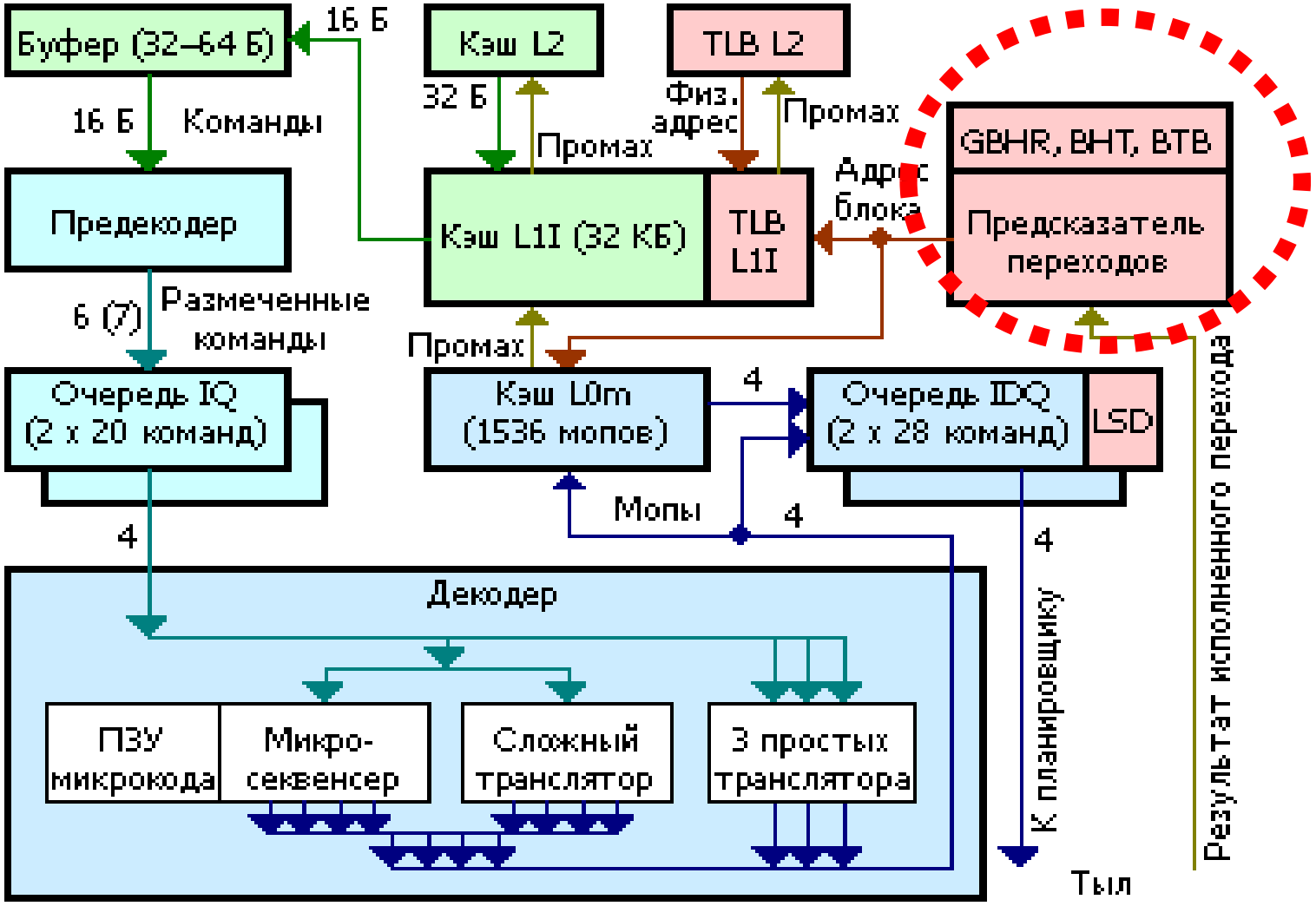


IP = адресная часть команды (CALL/RET)

или

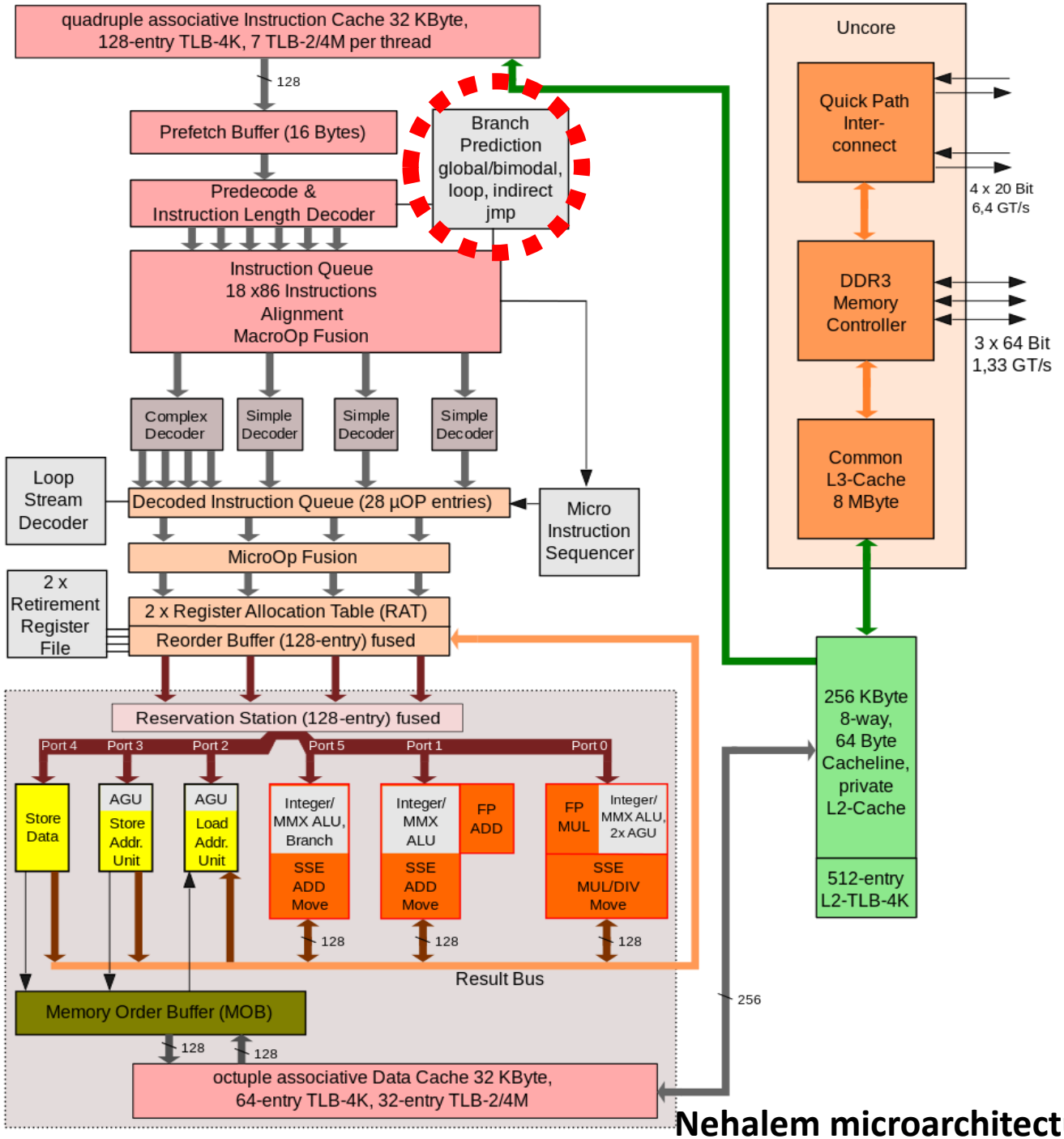
IP = IP + адресная часть команды (Jcc/LOOP)

БЛОКИ ПРЕДСКАЗАНИЯ ПЕРЕХОДОВ =
БЛОК ПРОГНОЗИРОВАНИЯ ВЕТВЛЕНИЙ



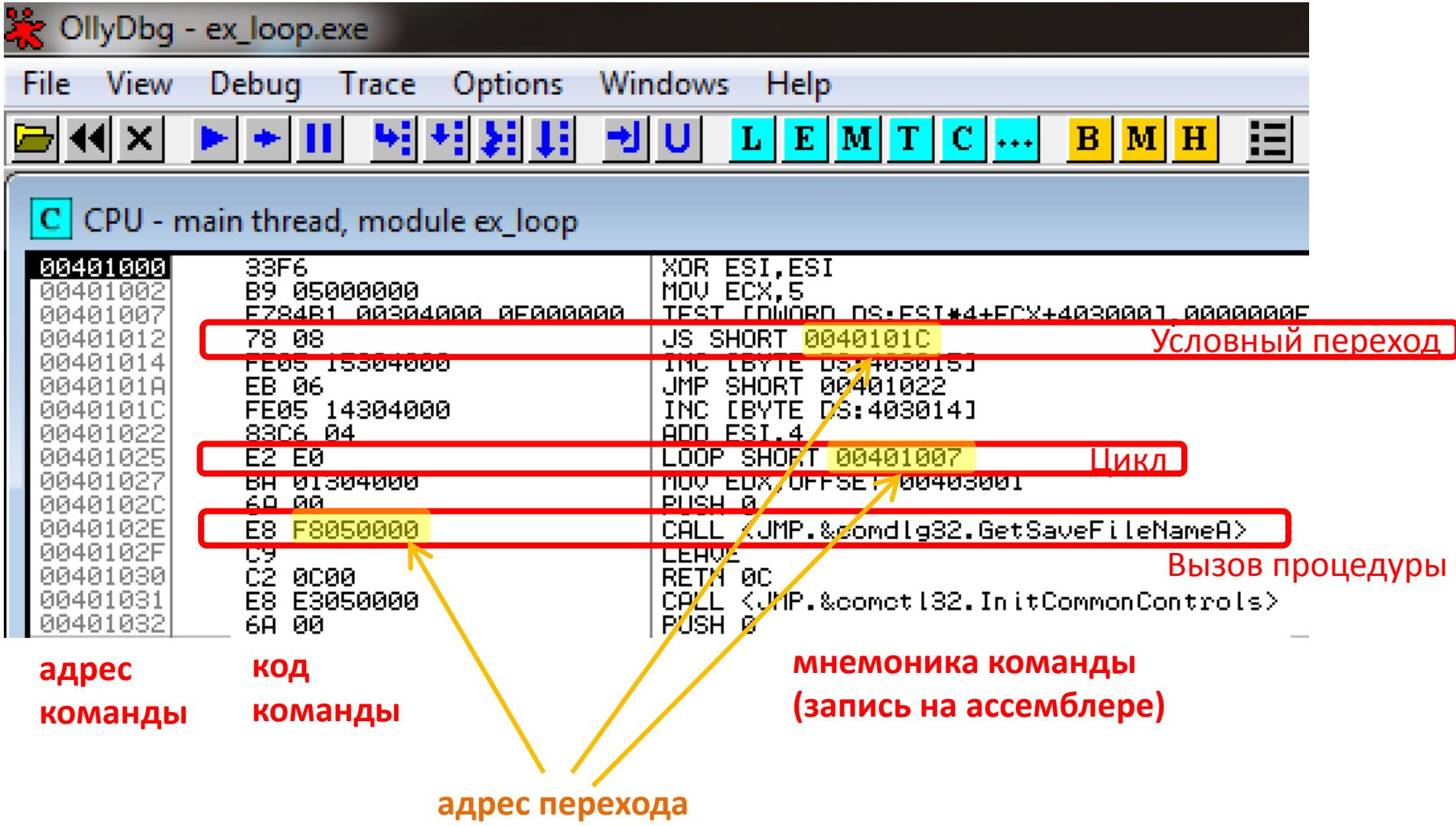
Sandy Bridge

Суперскалярная обработка. Аппаратная реализация



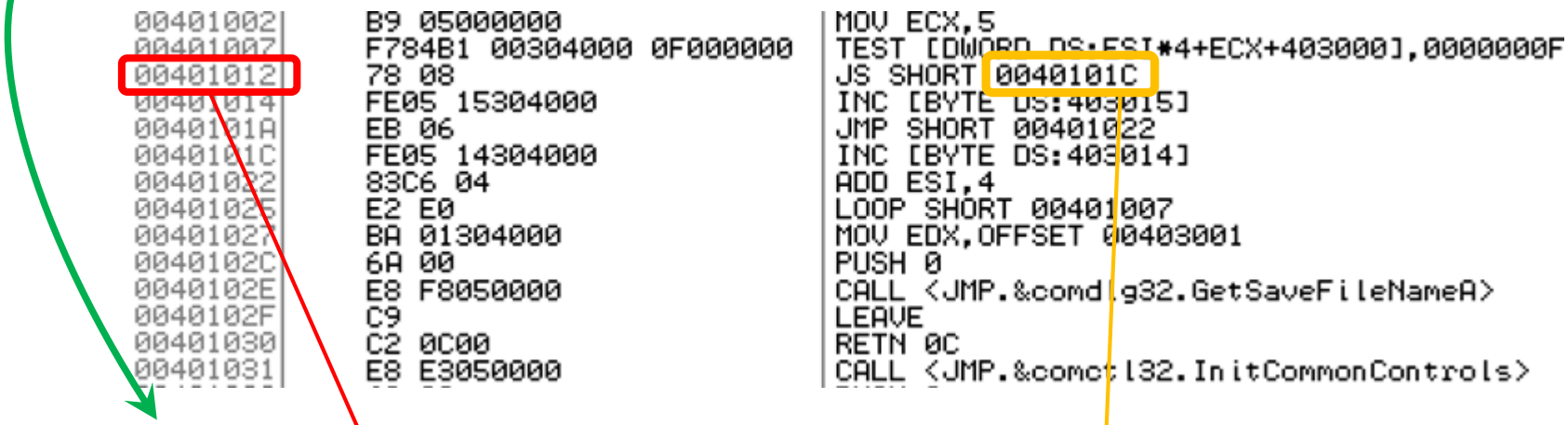
Nehalem microarchitecture

Пример кодов команд и их адресов



СТРУКТУРА БЛОКА ПРЕДСКАЗАНИЯ ПЕРЕХОДОВ

*Основой **BPU** (branch predictor unit) - **Блока предсказания переходов** является таблица, в которой запомнены адреса переходов, уже выполненных ранее.*



Адрес команды, являющейся командой передачи управления	Адрес выполненного перехода
00 40 10 12	00 40 10 1C
00 40 10 25	00 40 10 07
00 40 10 2E	00 00 03 FB

Если на момент чтения блока команд в нём встречается адрес, запомненный в таблице, то из таблицы выбирается адрес перехода, по которому будет выбираться следующий блок команд для запуска в конвейер.

АЛГОРИТМ ВЫПОЛНЕНИЯ ПЕРЕХОДА

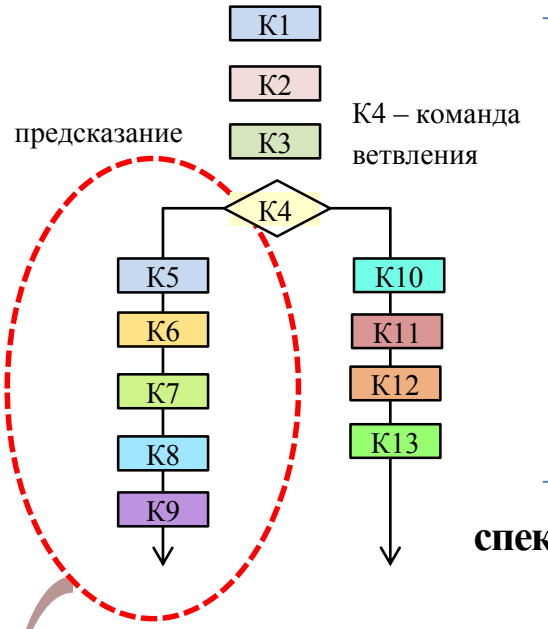
1. *Дешифратор прикрепляет к команде перехода оба адреса — 1)предсказанный адрес перехода и 2)предварительно признанный неудачным.*

2. *Выполняется предварительная выборка блока команд по предсказанному адресу. И одновременно целочисленный блок выполняет операцию перехода.*

3. *По окончании перехода определяется, какая из ветвей была выбрана.*

4. ***В случае перехода по предсказанию*** *все предварительно накопленные и выполненные команды данной ветви 1) маркируются как годные для дальнейшего использования, и продолжается исполнение данной ветви программы.*

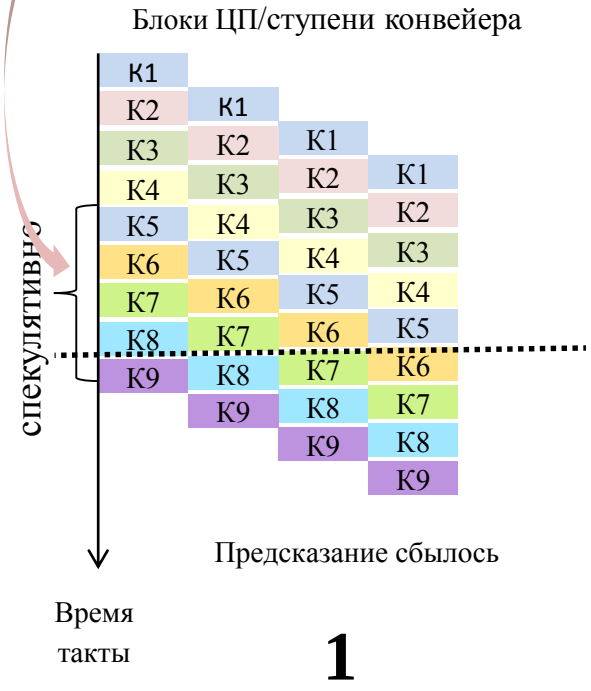
   В противном случае*, блок выполнения перехода (целочисленный блок) изменяет статус всех команд данной ветви на "подлежащие удалению". Потом передает в буфер адреса перехода правильный адрес перехода 2), и буфер, в свою очередь, перезапускает конвейер с этого адреса.*

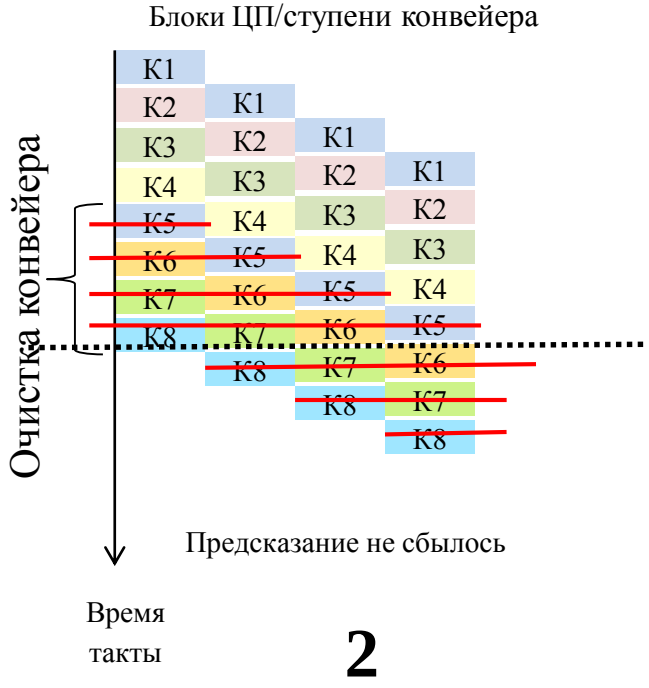


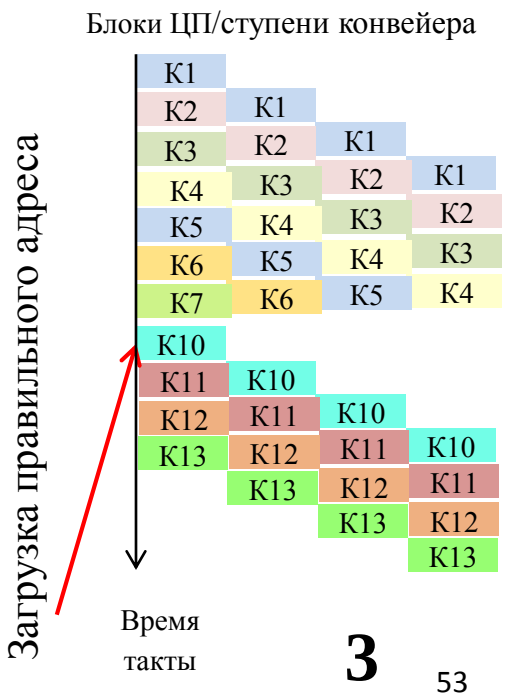
фрагмент программы

Порядок выполнения команд

спекулятивно(1), очистка конвейера(2), перезапуск конвейера с нового адреса(3)







АЛГОРИТМ ВЫПОЛНЕНИЯ ПЕРЕХОДА

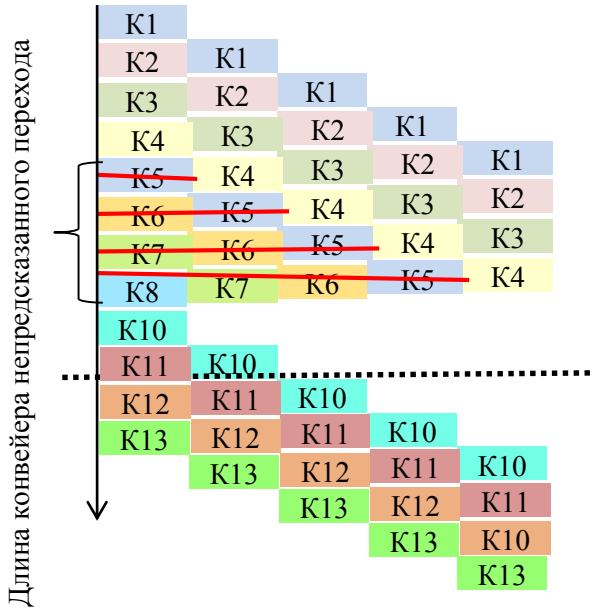
В момент отставки инструкции передачи управления (при неправильном предсказании перехода) все последующие инструкции загруженные в конвейер будут отменены, и начнётся считывание инструкций из I-кэша по правильному адресу. Такую процедуру называют СБРОСОМ КОНВЕЙЕРА, а время (в тактах), которое было потрачено на выполнение инструкции перехода с момента её считывания из кэша, называют ДЛИНОЙ КОНВЕЙЕРА НЕПРЕДСКАЗАННОГО перехода.

Это время характеризует чистую потерю в идеальных условиях, когда инструкция проходила через все этапы «гладко» и нигде не задерживалась по внешним причинам. В реальных условиях потеря на неправильно предсказанный переход может оказаться выше.

Длина конвейера непредсказанного перехода не всегда указывается в документации и известна весьма приблизительно. Её довольно трудно замерить, так как современные предсказатели переходов работают достаточно эффективно и не позволяют добиться гарантированной доли неправильных предсказаний в тестах.

Можно дать следующие примерные оценки длины конвейера (http://www.ixbt.com/cpu/cpu-microarchitecture-part-1.shtml):

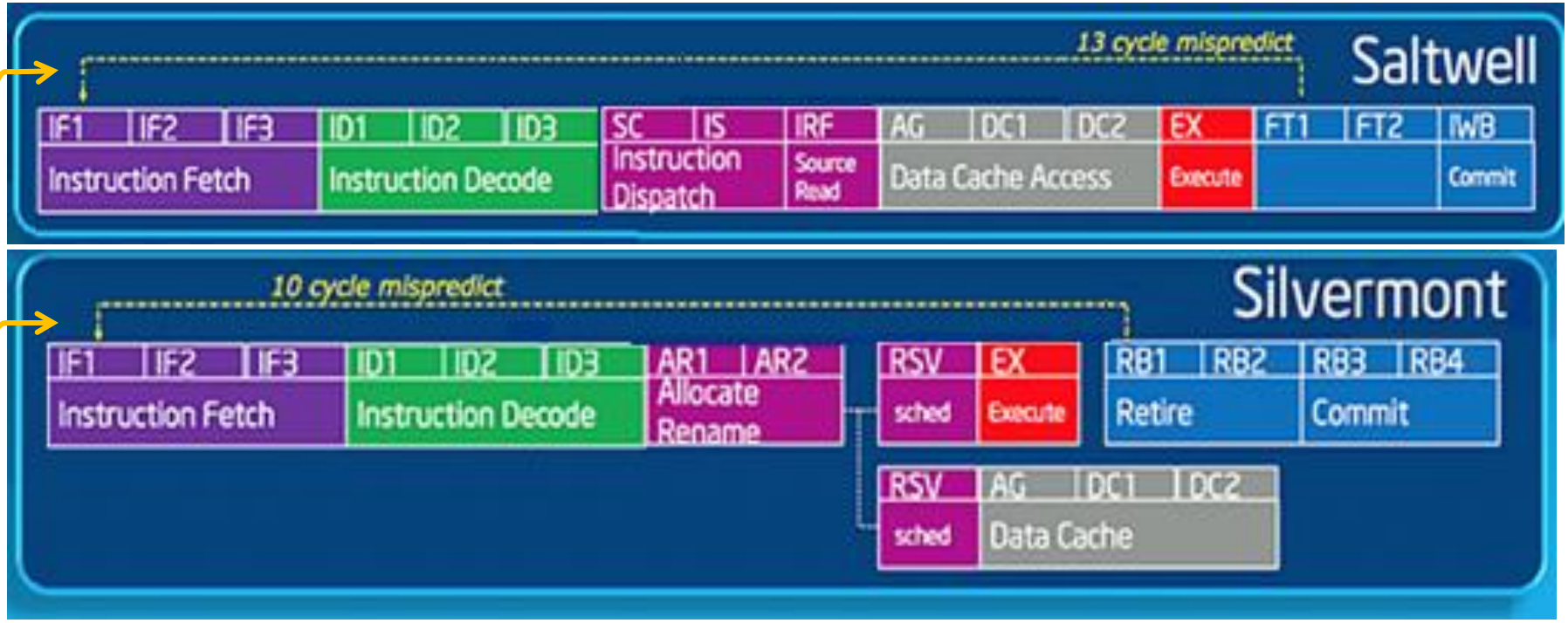
- Intel Pentium III (P6) — 11,
- Intel Core Duo (P-M) — 12,
- Intel Pentium 4 (P7) — 15,
- Intel Prescott P-4E — 22,
- Intel Core 2 Duo (P8) — 14,
- AMD Athlon 64 / Opteron (K8) — 11,
- IBM PowerPC 970 — 13.



СОВРЕМЕННЫЕ ПРОЦЕССОРЫ

с разбиением команды на 16 фаз/стадий

И ИХ ИЗДЕРЖКИ НЕПРАВИЛЬНЫХ ПРЕДСКАЗАНИЙ ПЕРЕХОДОВ



Длина конвейера непредсказанного перехода

http://www.ixbt.com/portopc/atom-clover5.shtml

(Планшеты на Intel Atom и Windows 8, часть 6. Новое в архитектуре Silvermont процессоров Intel Atom Z3xxx: что ждет нас в тонких, легких и недорогих планшетах на Windows 8 в 2014 году)

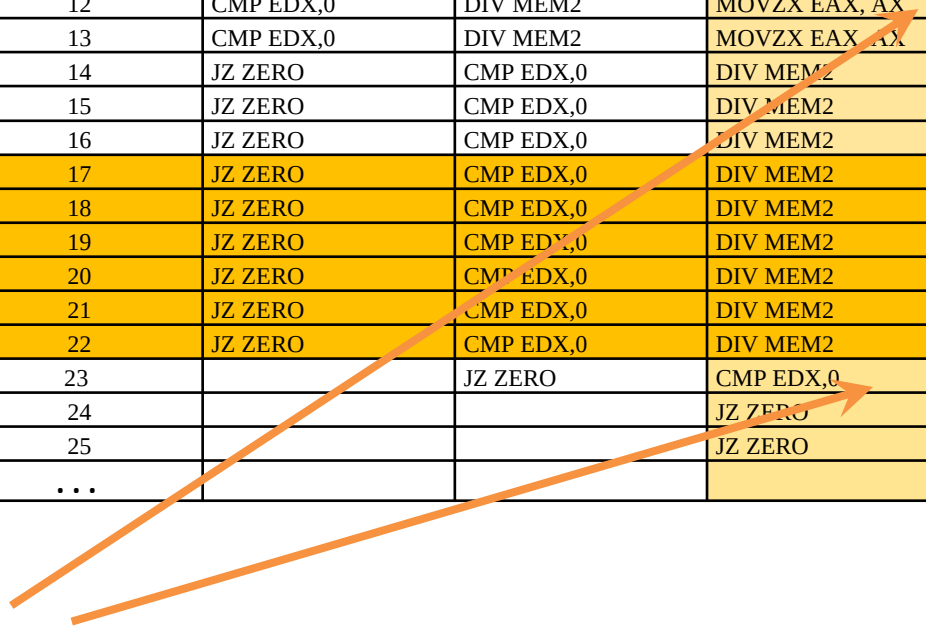
ПРОБЛЕМЫ/ЗАМЕДЛЕНИЕ ПРИ РАБОТЕ КОНВЕЙЕРА ЯДРА ПРОЦЕССОРА

- *Простой, вызванный различной длительностью разных фаз команд* √
- *Простой из-за зависимости команд по данным* √
- *Простой после команды передачи управления* √
- *Простой, вызванный КЭШ-промахом*

Простои в работе конвейера команд

SUB EDX, EDX
MOV AX, **MEM1**
MOVZX EAX, AX
DIV **MEM2**
CMP EDX,0
JZ ZERO
PUSH **STR1**

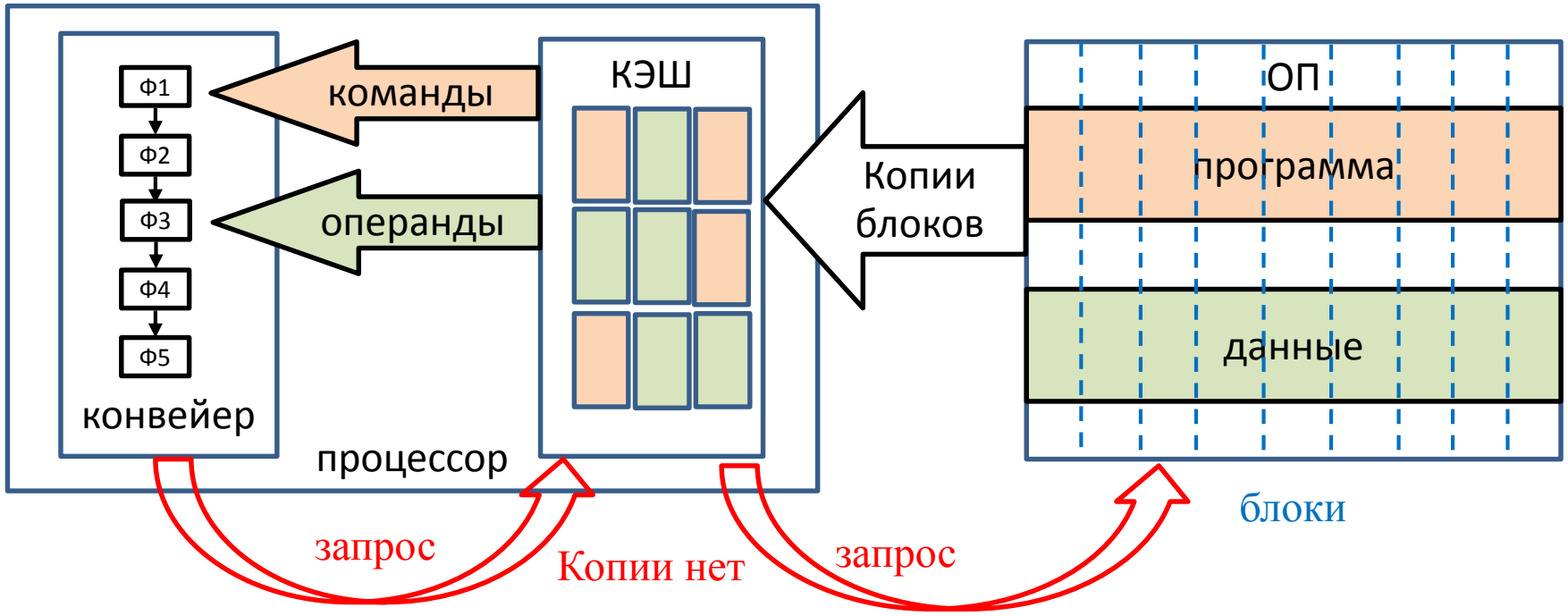
Время (такты)	Выборка команды	Декодирование	Выборка операндов	Исполнение операции	Запись результата
1	SUB EDX, EDX				
2	MOV AX, MEM1	SUB EDX, EDX			
3	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX		
4	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1	SUB EDX, EDX	
5	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		SUB EDX, EDX
6	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
7	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
8	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
9	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
10	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
11	DIV MEM2	MOVZX EAX, AX	MOV AX, MEM1		
12	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		MOV AX, MEM1
13	CMP EDX,0	DIV MEM2	MOVZX EAX, AX		
14	JZ ZERO	CMP EDX,0	DIV MEM2		MOVZX EAX, AX
15	JZ ZERO	CMP EDX,0	DIV MEM2		
16	JZ ZERO	CMP EDX,0	DIV MEM2		
17	JZ ZERO	CMP EDX,0	DIV MEM2		
18	JZ ZERO	CMP EDX,0	DIV MEM2		
19	JZ ZERO	CMP EDX,0	DIV MEM2		
20	JZ ZERO	CMP EDX,0	DIV MEM2		
21	JZ ZERO	CMP EDX,0	DIV MEM2		
22	JZ ZERO	CMP EDX,0	DIV MEM2		
23		JZ ZERO	CMP EDX,0	DIV MEM2	
24			JZ ZERO	CMP EDX,0	DIV MEM2
25			JZ ZERO		CMP EDX,0
. . .					



Простой из-за КЭШ-промаха (копий искомых данных нет в ОП)

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО КЭШ-ПРОМАХАМИ

Команда К1 при чтении данных из КЭШ не нашла там достоверной копии, значит надо обратиться в ОП. Это в несколько раз дольше, чем прочитать из КЭШ. Все последующие команды простаивают.



Предложите способы борьбы с этими простоями

? ? ? ? ?

СПОСОБЫ УСТРАНЕНИЯ ПРОСТОЯ, ВЫЗВАННОГО КЭШ-ПРОМАХАМИ

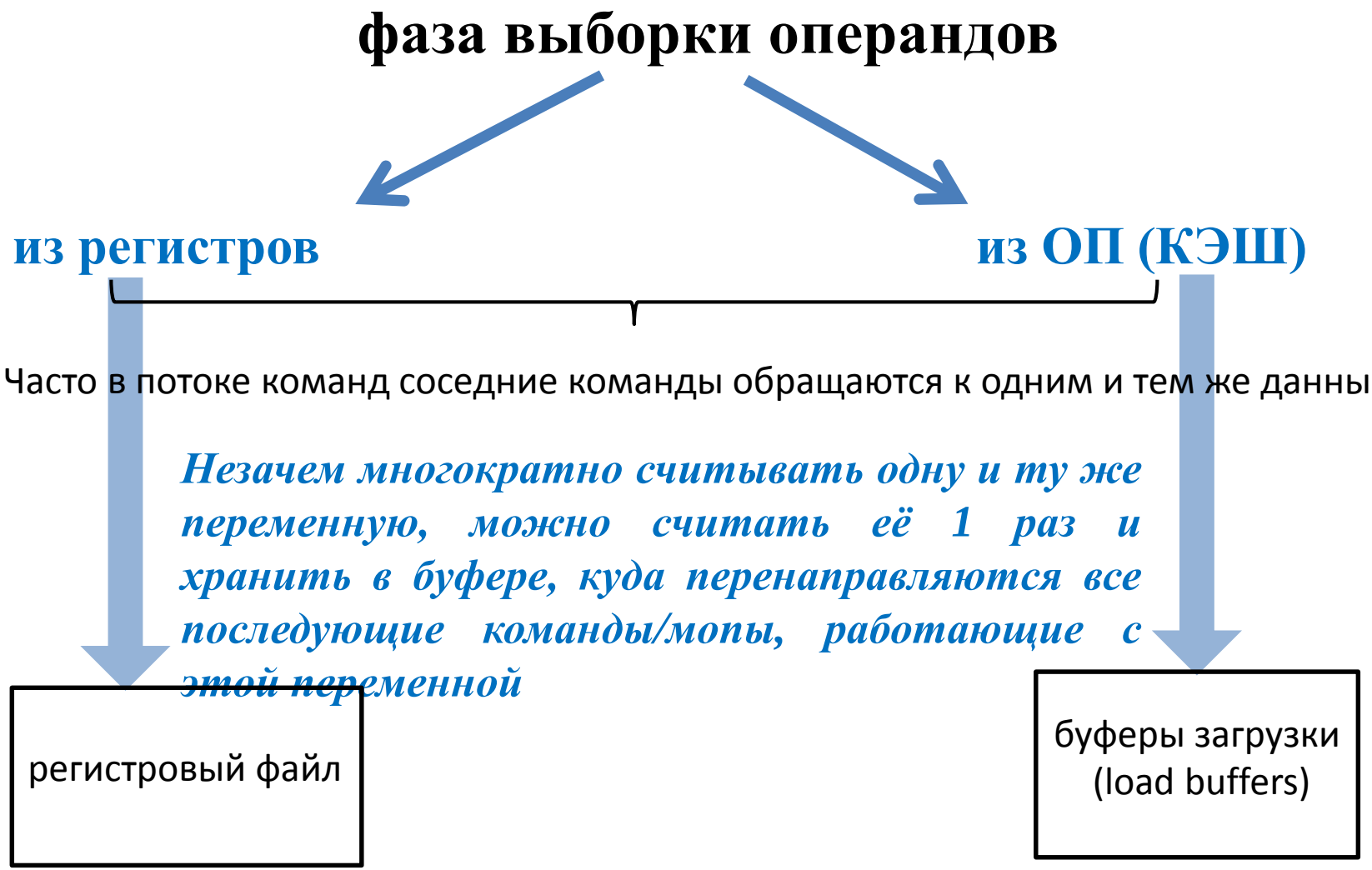
Команда при чтении данных из КЭШ не нашла там достоверной копии блока ОП, значит надо обратиться за оригиналом в ОП. Это в несколько раз дольше, чем прочитать из КЭШ. Все последующие команды простаивают.

Поможет

- *внеочередное исполнение*
- *временное хранение последних результатов по программе во внутренних буферах процессора*
- *предсказание адресов данных и предварительная выборка данных из ОП*

Временное хранение операндов и результатов

= использование множества буферов загрузки/сохранения



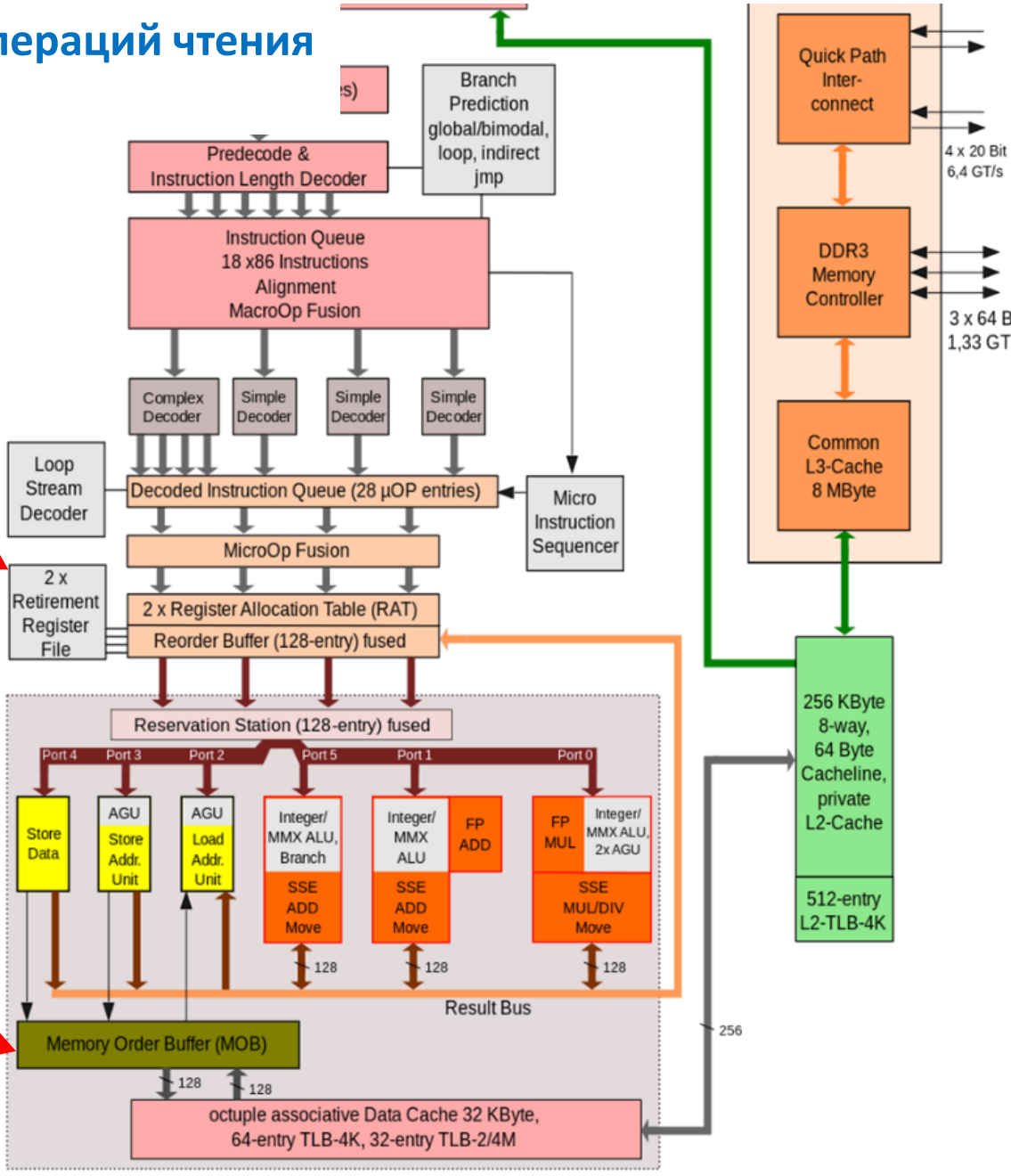
* - http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf

БЛОКИ ВЫБОРКИ ДАННЫХ (ОПЕРАНДОВ)

- исключение дублирующих операций чтения
- использование буферов

регистровый файл

буфер загрузки/записи



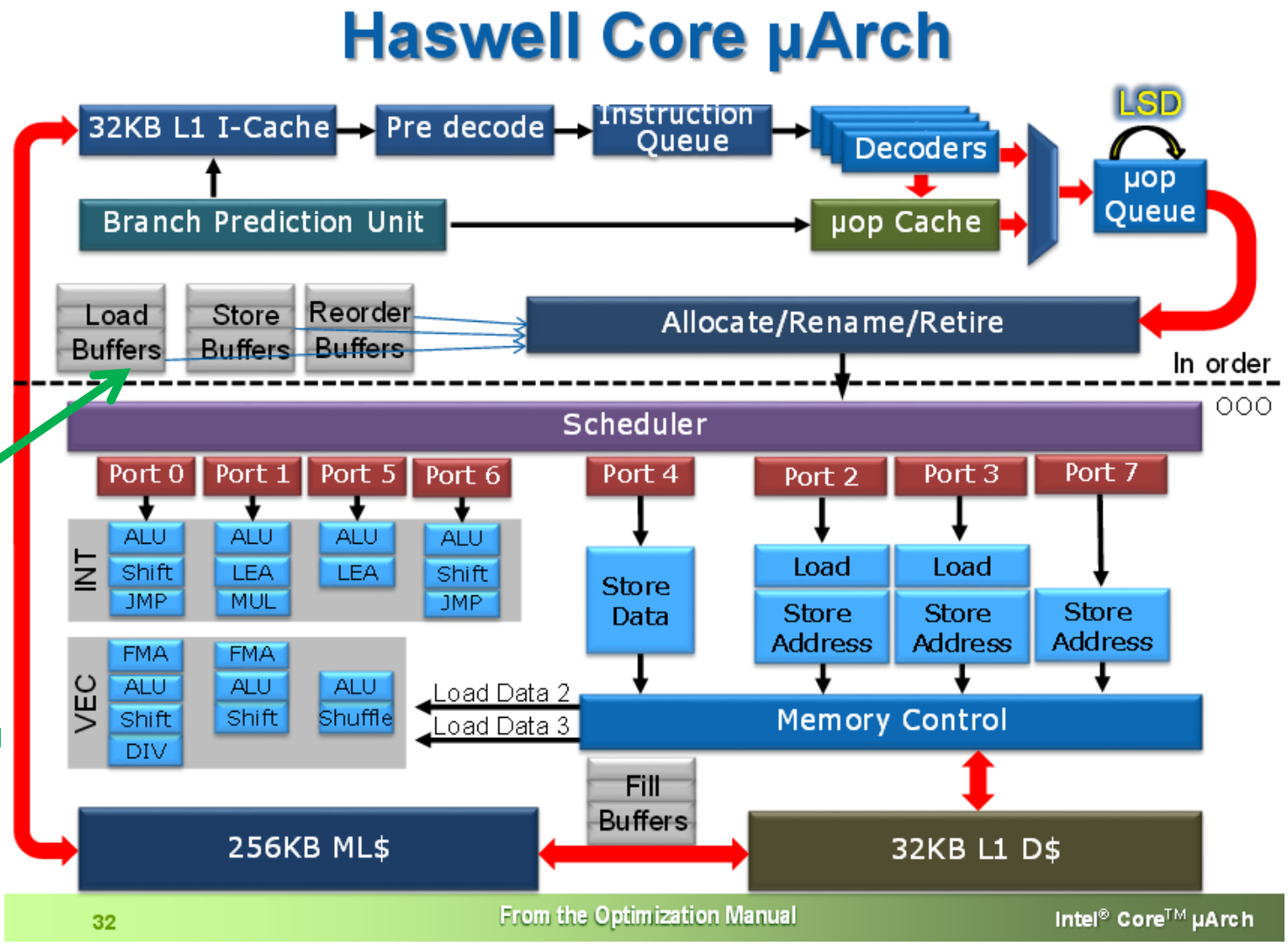
БЛОКИ ВЫБОРКИ ДАННЫХ (ОПЕРАНДОВ)

- исключение дублирующих операций чтения
- использование буферов

буферы загрузки

и

регистровый файл



ПРЕДСКАЗАНИЕ АДРЕСОВ ДАННЫХ

УСТРОЙСТВА ПРЕДВАРИТЕЛЬНОЙ ВЫБОРКИ ДАННЫХ (из КЭШ/ОП)

потоковой предвыборки

Алгоритм предугадывания:

- принимается большая вероятность линейности запросов к обрабатываемым данным (элементов массивов, строк, таблиц…)
- в КЭШ предварительно загружается ближайший к текущему блоку ОП

пошаговой предвыборки

Алгоритм предугадывания:

базируется на регистре-указателе команд (IP)

- отслеживает отдельные инструкции загрузки (чтения данных из ОП), вычисляя постоянный шаг (в цикле)
- адрес упреждающей выборки = текущий адрес + шаг

http://moodle.technion.ac.il/pluginfile.php/313428/mod_resource/content/1/Core%20uArch%20-%20Advanced%20VLSI%20Arch.pdf