



# **АРХИТЕКТУРА И МИКРОАРХИТЕКТУРА В ПРОЦЕССОРАХ И СИСТЕМАХ НА КРИСТАЛЛЕ**

ДОКТОР ТЕХНИЧЕСКИХ НАУК, ПРОФЕССОР ПАЛТАШЕВ ТИМУР ТУРСУНОВИЧ  
ADVANCED MICRO DEVICES, RADEON TECHNOLOGY GROUP  
AUGUST 25, 2016

## АРХИТЕКТУРА И МИКРОАРХИТЕКТУРА В ПРОЦЕССОРАХ И СИСТЕМАХ НА КРИСТАЛЛЕ

### Определение архитектуры

Базовая архитектура процессора фон Неймана и основные компоненты компьютера

### Архитектура системы команд

Complex instruction Set Computer (CISC)  
Reduced Instruction Set Computer (RISC)

### Особенности системы команд

Принципы исполнения команд в микропроцессоре: что нужно сделать, чтобы получить результат

### Микроархитектура в базовых блоках

Микроархитектуры, управление и компоненты, реализующие исполнение команд: последовательные и конвейерные

### Все возможные излишества

Микроархитектуры для повышения производительности: суперскалярные, VLIW, многотредные и прочие

# КЛАССЫ МИКРОПРОЦЕССОРОВ



МИКРОПРОЦЕССОРОВ В МИРЕ ГОРАЗДО БОЛЬШЕ, ЧЕМ ЛЮДЕЙ И ЖИВОТНЫХ,  
И ИХ ЧИСЛО ПОСТОЯННО РАСТЕТ

**Дискретные  
микроспроцессоры  
CPU (отдельный  
кристалл)**

**Встроенные  
микроспроцессоры  
CPU (ядро для  
системы на  
кристалле)**

**Графические  
микроспроцессоры  
GPU (дискретные и  
встроенные)**

**Микроспроцессоры  
обработки сигналов  
DSP (дискретные и  
встроенные)**

**Аудио и видео-  
процессоры,  
микроспроцессоры  
обработки  
изображений**

**Реконфигуриру-  
емые FPGA  
микроспроцессоры  
для прототипов и  
спецзадач**

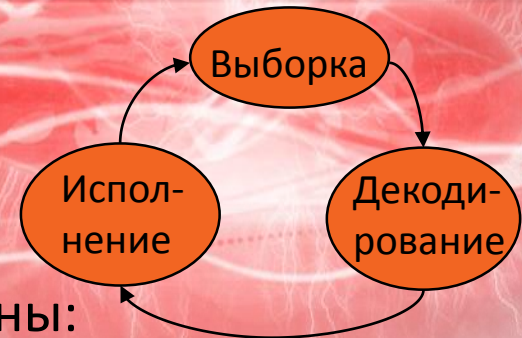
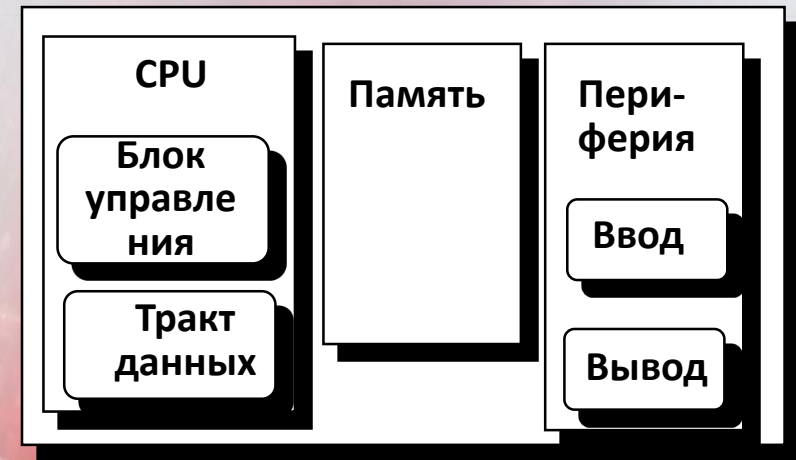
**Гетерогенные мультипроцессоры**

# БАЗОВАЯ АРХИТЕКТУРА ПРОЦЕССОРА ФОН НЕЙМАНА

ЯВЛЯЕТСЯ ПРОТОТИПОМ БОЛЬШИНСТВА СОВРЕМЕННЫХ МИКРОПРОЦЕССОРОВ

▲ **Блок управления** должен обеспечивать:

1. Выборку команд из памяти
2. Инициировать сигналы управления для информационного потока между компонентами тракта данных, а также управлять типами исполняемых операций
3. Управлять последовательностью исполнения команд



▲ В состав тракта данных должны быть включены:

- ❑ **Компоненты** – функциональные блоки, регистры и модули памяти, которые необходимы для исполнения команд
- ❑ **Межсоединения** – компоненты должны быть соединены таким образом, чтобы команды могли быть исполнены и все необходимые данные могли быть загружены из памяти и затем туда мог быть записан результат



# АРХИТЕКТУРА СИСТЕМЫ КОМАНД МИКРОПРОЦЕССОРА

## РАЗНЫЕ СТИЛИ УПРАВЛЕНИЯ ЧЕРЕЗ СИСТЕМУ КОМАНД

### ▲ **Complex instruction Set Computer (CISC)**

- Архитектура процессора, в котором одна команда может выполнять несколько операций (таких как загрузка из памяти с последующими арифметической операцией и записью в память результата) или выполняющей многошаговые операции с различными режимами адресации
- Множество сложных команд с многими форматами/размерами команды
- Характерна для 1960-70х годов ввиду многотактовости исполнения команд

### ▲ **I8086 и его последователи x86 от Intel и AMD ( есть также VIA Technologies)**

### ▲ **Reduced Instruction Set Computer (RISC)**

- Подход к построению процессора, в котором использован факт, что упрощенная система команд обеспечивает более высокую производительность в случае комбинирования с микроархитектурой, позволяющей исполнять эти команды за меньшее число тактов
- 32/64-бит архитектура типа Load/Store, отделяющая арифметические операции
- Большинство команд исполняется с единой задержкой в несколько тактов и каждый такт начинает исполняться новая команда в режиме конвейера
- Формат большинства команд унифицированный (обычный 32-бита и сжатый 16 бит)

### ▲ **MIPS, ARM, SPARC, IBM PowerPC, HP-RISC etc.**

# АРХИТЕКТУРА СИСТЕМА КОМАНД MIPS R3000



## ТИПЫ И ФОРМАТЫ КОМАНД, НАБОР РЕГИСТРОВ ВИДИМЫХ ПРОГРАММИСТУ

### ▲ Типы команд процессора

- Вычислительные
- Загрузки/Записи из/в память
- Перехода и ветвления
- Операции с плавающей точкой
  - сопроцессор
- Управление памятью
- Специальные команды

Регистры

R0 - R31

Счетчик команд PC

HI Верхний

LO Нижний

3 формата команд: все шириной 32 бита

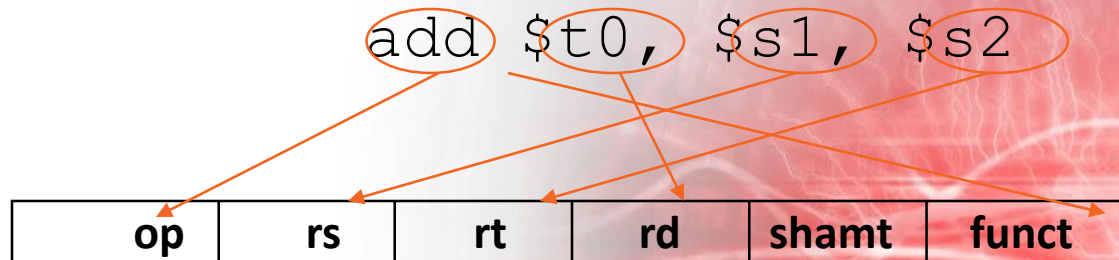
OP	rs	rt	rd	sa	funct	R формат
OP	rs	rt	immediate			I формат
OP	jump target					J формат

# МАШИННЫЙ ЯЗЫК – КОМАНДА СЛОЖЕНИЯ В MIPS



## КАКИМ ОБРАЗОМ ПРОЦЕССОР ПОНИМАЕТ СУТЬ КОМАНДЫ

- ▲ Машинная команда: «Где взять, Что сделать, Куда положить»
- ▲ Все команды, как и регистры имеют размер в 32-бита
- ▲ Пример арифметической команды сложения в R формате:



op	6-bits	opcode определяет операцию
rs	5-bits	адрес первого операнда в регистровом файле
rt	5-bits	адрес второго операнда в регистровом файле
rd	5-bits	адрес результата в регистровом файле
shamt	5-bits	Число позиций для команды сдвига <b>shift amount</b>
funct	6-bits	Дополнительный код операции <b>function code</b>



# АРХИТЕКТУРА СИСТЕМЫ КОМАНД: РЕЖИМЫ АДРЕСАЦИИ

## КАКИМ ОБРАЗОМ ПРОЦЕССОР ОБЕСПЕЧИВАЕТ ДОСТУП К ДАННЫМ ИЗ ПАМЯТИ

1. **Регистровая адресация:** Операнд находится в одном из регистров.
2. **Непосредственная адресация (или константа):** операнд является частью команды
3. **Прямая или абсолютная адресация:** адрес операнда в памяти является частью команды
4. **Косвенная регистровая адресация:** адрес операнда в памяти хранится в одном из регистров
5. **Автоинкрементная (или косвенная регистровая с постинкрементом):** тоже самое, что 4; но содержание регистра инкрементируется после использования адреса
6. **Автодекрементная (или косвенная регистровая с преддекрементом):** тоже самое, что 5; но содержание регистра декрементируется перед использованием адреса
7. **Косвенная регистровая (базовая) со смещением:** адрес операнда вычисляется как сумма содержимого базового регистра и 8- или 16-бит смещения
8. **Косвенная базовая индексная адресация:** работает также как 7; адрес является суммой содержимого базового регистра и индексного регистра, содержание которого может быть масштабировано с фактором 1,2,4,8 или 16
9. **Косвенная базовая индексная адресация со смещением:** работает также как 8; только еще добавляется смещение для формирования окончательного адреса
10. **Относительная адресация по счетчику команд:** смещение добавляется к значению счетчика команд; используется для команд перехода и ветвления.



# ПРИМЕР НЕКОТОРЫХ КОМАНД MIPS



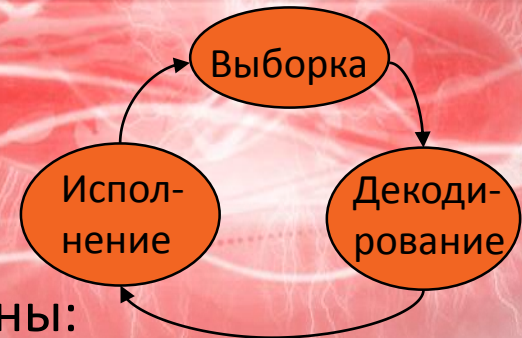
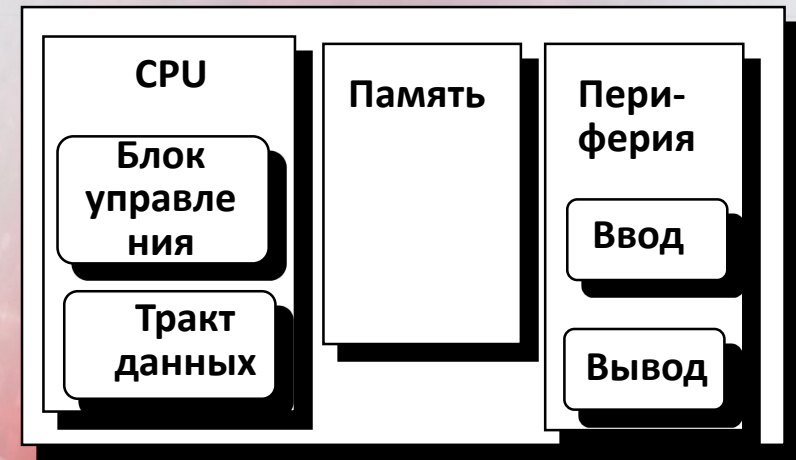
Category	Instr	Op Code	Example	Meaning
Arithmetic (R & I format)	add	0 and 32	add \$s1, \$s2, \$s3	$\$s1 = \$s2 + \$s3$
	subtract	0 and 34	sub \$s1, \$s2, \$s3	$\$s1 = \$s2 - \$s3$
	add immediate	8	addi \$s1, \$s2, 6	$\$s1 = \$s2 + 6$
	or immediate	13	ori \$s1, \$s2, 6	$\$s1 = \$s2 \vee 6$
Data Transfer (I format)	load word	35	lw \$s1, 24(\$s2)	$\$s1 = \text{Memory}(\$s2+24)$
	store word	43	sw \$s1, 24(\$s2)	$\text{Memory}(\$s2+24) = \$s1$
	load byte	32	lb \$s1, 25(\$s2)	$\$s1 = \text{Memory}(\$s2+25)$
	store byte	40	sb \$s1, 25(\$s2)	$\text{Memory}(\$s2+25) = \$s1$
	load upper imm	15	lui \$s1, 6	$\$s1 = 6 * 2^{16}$
Cond. Branch (I & R format)	br on equal	4	beq \$s1, \$s2, L	if ( $\$s1 == \$s2$ ) go to L
	br on not equal	5	bne \$s1, \$s2, L	if ( $\$s1 \neq \$s2$ ) go to L
	set on less than	0 and 42	slt \$s1, \$s2, \$s3	if ( $\$s2 < \$s3$ ) $\$s1=1$ else $\$s1=0$
	set on less than immediate	10	slti \$s1, \$s2, 6	if ( $\$s2 < 6$ ) $\$s1=1$ else $\$s1=0$
Uncond. Jump (J & R format)	jump	2	j 2500	go to 10000
	jump register	0 and 8	jr \$t1	go to \$t1
	jump and link	3	jal 2500	go to 10000; $\$ra=PC+4$

# БАЗОВАЯ АРХИТЕКТУРА ПРОЦЕССОРА ФОН НЕЙМАНА **AMD**

ЯВЛЯЕТСЯ ПРОТОТИПОМ БОЛЬШИНСТВА СОВРЕМЕННЫХ МИКРОПРОЦЕССОРОВ

▲ **Блок управления** должен обеспечивать:

1. Выборку команд из памяти
2. Инициировать сигналы управления для информационного потока между компонентами тракта данных, а также управлять типами исполняемых операций
3. Управлять последовательностью исполнения команд



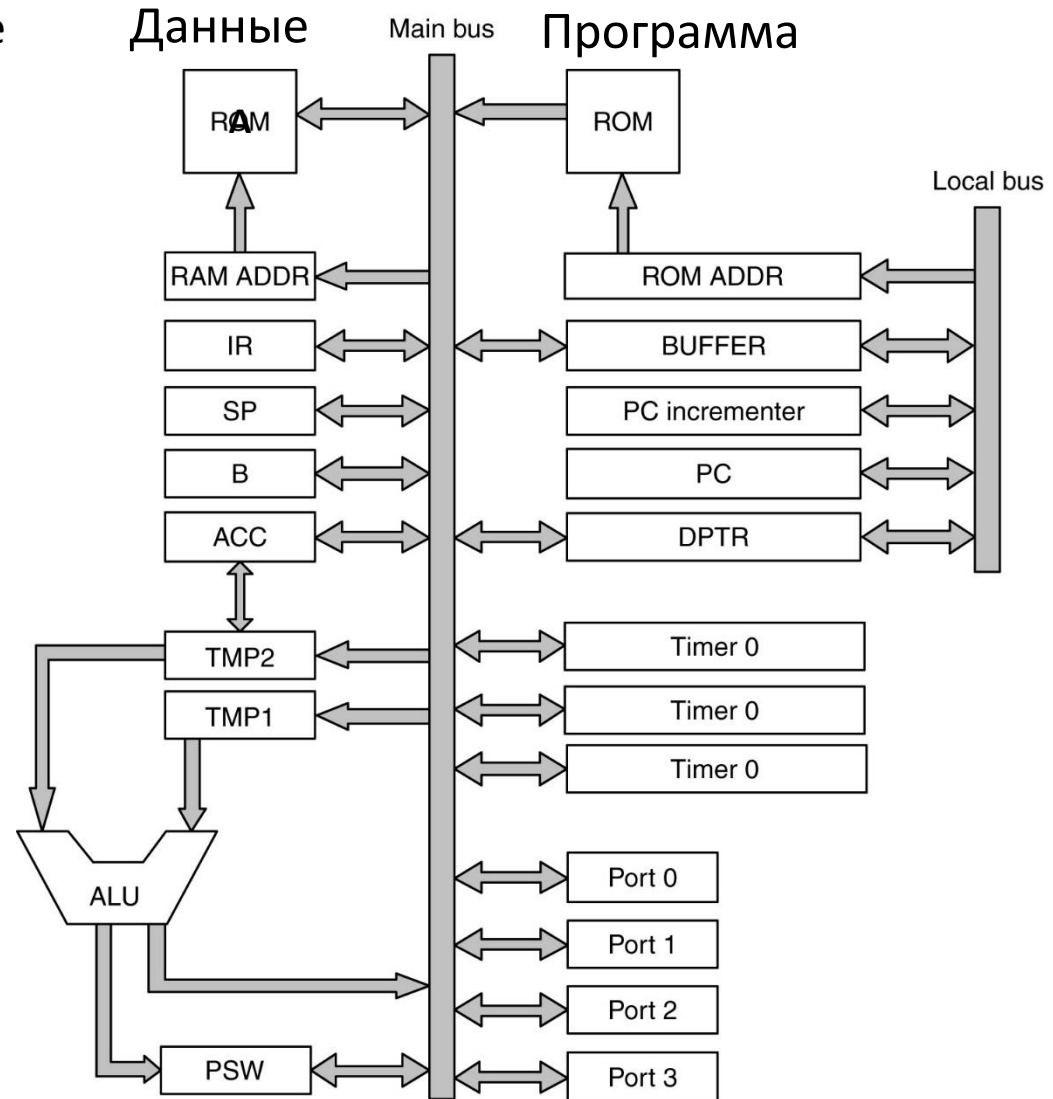
▲ В состав тракта данных должны быть включены:

- ❑ **Компоненты** – функциональные блоки, регистры и модули памяти, которые необходимы для исполнения команд
- ❑ **Межсоединения** – компоненты должны быть соединены таким образом, чтобы команды могли быть исполнены и все необходимые данные могли быть загружены из памяти и затем туда мог быть записан результат

# МИКРОАРХИТЕКТУРА МИКРОКОНТРОЛЛЕРА INTEL 8051

## ГАРВАРДСКАЯ АРХИТЕКТУРА: ПРОГРАММА И ДАННЫЕ РАЗНЕСЕНЫ В ROM И RAM

- ▲ Два блока памяти и периферийные устройства в дополнение к тракту данных и блоку управления
  - Таймеры и внешние порты
  - Постоянная память программ ROM и память данных произвольного доступа RAM
  - Соединены через главную и локальные шины
- ▲ Регистры блока управления
  - Регистры команды IR и счетчика команд PC с инкрементом
  - Регистры адресов памяти RAM/ROM, DPTR и указателя вершины стека SP
  - Регистр состояния программы PSW
- ▲ Регистры и АЛУ тракта данных
  - Регистр-аккумулятор ACC и временные регистры операндов TMP1/2
  - В-регистр для промежуточных рез-в
  - АЛУ для базовых операций с 8-бит данными





# ПРИМЕР МИКРОАРХИТЕКТУРЫ ТРАКТА ДАННЫХ

TANENBAUM, STRUCTURED COMPUTER ORGANIZATION. FIFTH EDITION. (C) 2006

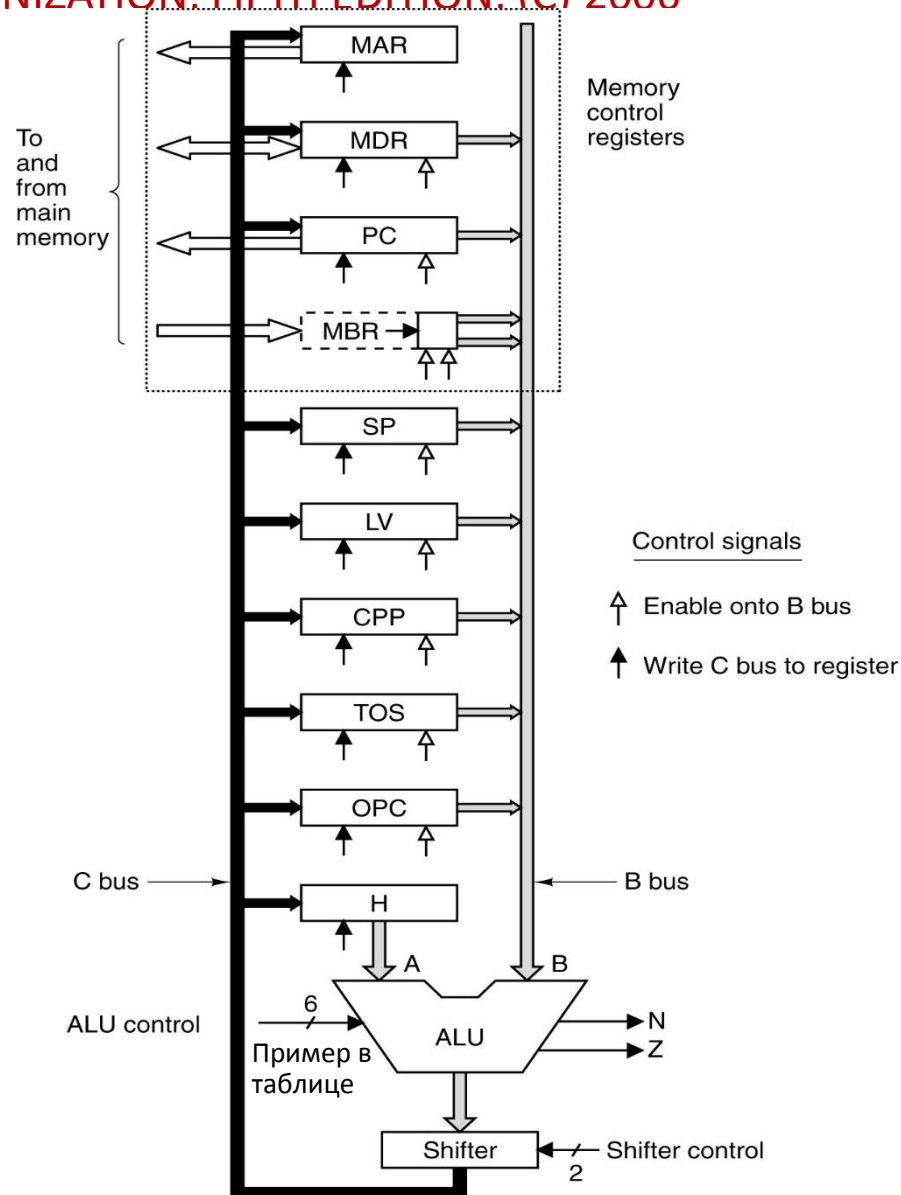


## ▲ Основные элементы тракта данных

- Регистры данных, адреса, стека и состояния
- Межсоединения в виде шин или магистралей В и С, а также шины доступа к памяти
- Функциональные блоки в виде АЛУ и сдвигателя
- Наборы управляющих сигналов для записи данных из шины С в регистры и из регистров на шину В
- Наборы управляющих сигналов для управления АЛУ и сдвигом
- Сигналы флагов результата операций N, Z с соответствующим регистром состояния

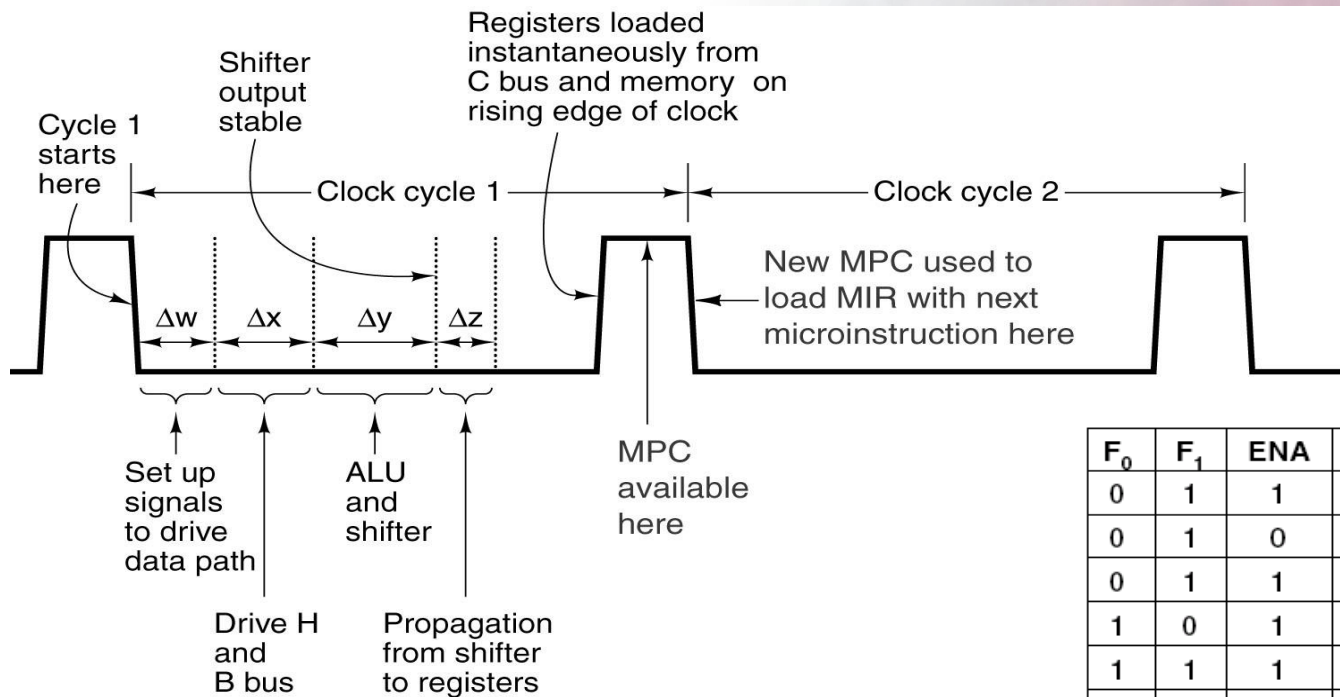
## ▲ Конвейерная микроархитектура в лекциях Imagination Technologies

- MIPS pipelined implementation



# ПРИМЕР ВРЕМЕННОЙ ДИАГРАММЫ И УПРАВЛЕНИЯ АЛУ **AMD**

## ВСЕ ПРОБЛЕМЫ МИКРОАРХИТЕКТУРЫ В ЗАДЕРЖКАХ, СИГНАЛАХ И СИНХРОНИЗАЦИИ



F <sub>0</sub>	F <sub>1</sub>	ENA	ENB	INVA	INC	Function
0	1	1	0	0	0	A
0	1	0	1	0	0	B
0	1	1	0	1	0	$\bar{A}$
1	0	1	1	0	0	$\bar{B}$
1	1	1	1	0	0	A + B
1	1	1	1	0	1	A + B + 1
1	1	1	0	0	1	A + 1
1	1	0	1	0	1	B + 1
1	1	1	1	1	1	B - A
1	1	0	1	1	0	B - 1
1	1	1	0	1	1	-A
0	0	1	1	0	0	A AND B
0	1	1	1	0	0	A OR B
0	1	0	0	0	0	0
1	1	0	0	0	1	1
1	1	0	0	1	0	-1

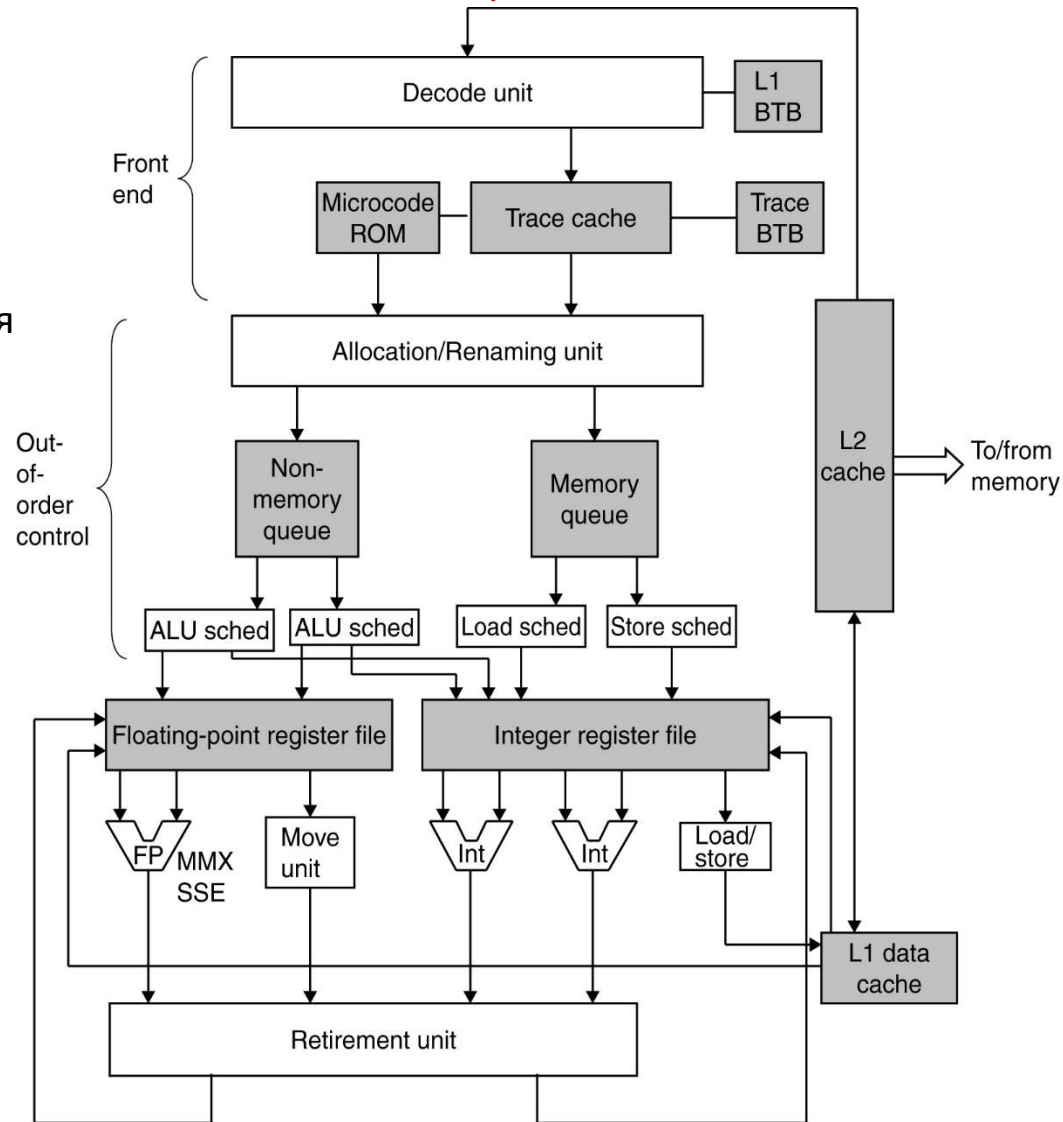
- ▲ Потактовое исполнение с изменением статуса в периоде между тактами исполнения команды (однотактное или многотактное)
- ▲ Каждое устройство или шина имеет задержку, суммарная задержка не должна превышать период
- ▲ Управляющие сигналы для устройств и шин должны подаваться в начале такта (задний фронт)

# УПРОЩЕННАЯ МИКРОАРХИТЕКТУРА PENTIUM 4



## СОВРЕМЕННЫЕ МИКРОПРОЦЕССОРЫ С МНОЖЕСТВОМ ФУНКЦИОНАЛЬНЫХ БЛОКОВ

- Многоступенчатая обработка каждой команды с момента начала выборки из памяти
  - Называется конвейером или *pipelined execution*
  - Аллокация внутренних регистров для операндов и результатов
- Несколько параллельных конвейеров для различных типов команд
  - Очереди и диспетчеризация исполнения в зависимости от доступности входных данных
  - Отдельные регистровые файлы и АЛУ для целых и вещественных операндов, выборки/записи в память
  - Несколько уровней кэш-памяти и специальные кэш-буферы для предсказания команд переходов
  - Специальный блок окончательного завершения команд





# ПОВЫШЕНИЕ ПРОИЗВОДИТЕЛЬНОСТИ ПРОЦЕССОРОВ

## ГЛАВНЫЕ АРХИТЕКТУРНЫЕ ПРИНЦИПЫ ДЛЯ РОСТА ПРОИЗВОДИТЕЛЬНОСТИ

### ▲ **Исполнение команд вне порядка их следования в машинном коде**

- Несколько независимых по данным команд одной программы могут исполняться одновременно при наличии нескольких трактов данных
- Строго последовательно команды исполняются при наличии только одного тракта данных или взаимной зависимости по данным

### ▲ **Использование скалярного конвейера и переход к суперскалярному конвейеру**

- Скалярный конвейер позволяет получать один 32/64-бит результат за такт, выполняя одновременно несколько команд, находящихся на разных ступенях
- Суперскалярный конвейер позволяет получать несколько 32/64-бит результатов за такт, используя параллельные тракты данных и функциональные блоки (P4)

### ▲ **Использование векторных процессоров для многокомпонентных данных**

- От суперкомпьютеров и мощных графических станций до фотокамер и телефонов

### ▲ **Использование многоядерных процессоров**

- Уменьшение размеров транзисторов до 7-10 нм позволяет построить 8-64 ядерные процессоры на одном кристалле и многократно увеличить производительность за счет параллелизма вычислительных процессов

### ▲ **Использование многотредных процессоров**

- Несколько аппаратно поддерживаемых тредов позволяют повысить загрузку трактов данных, избегая простоев в ожидании данных из памяти при одном треде

## ▲ Вопросы и ответы

### Контакты:

- ▲ Тимур Турсунович Палташев, Timour Paltashev
  - Старший Менеджер, Senior Manager
  - Advanced Micro Devices
  - Radeon Technology Group
  - GPU architecture and global academic connections
  - +1 408 306 8508
  - [timour.paltashev@amd.com](mailto:timour.paltashev@amd.com)

# DISCLAIMER & ATTRIBUTION



The information presented in this document is for informational purposes only and may contain technical inaccuracies, omissions and typographical errors.

The information contained herein is subject to change and may be rendered inaccurate for many reasons, including but not limited to product and roadmap changes, component and motherboard version changes, new model and/or product releases, product differences between differing manufacturers, software changes, BIOS flashes, firmware upgrades, or the like. AMD assumes no obligation to update or otherwise correct or revise this information. However, AMD reserves the right to revise this information and to make changes from time to time to the content hereof without obligation of AMD to notify any person of such revisions or changes.

AMD MAKES NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE CONTENTS HEREOF AND ASSUMES NO RESPONSIBILITY FOR ANY INACCURACIES, ERRORS OR OMISSIONS THAT MAY APPEAR IN THIS INFORMATION.

AMD SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL AMD BE LIABLE TO ANY PERSON FOR ANY DIRECT, INDIRECT, SPECIAL OR OTHER CONSEQUENTIAL DAMAGES ARISING FROM THE USE OF ANY INFORMATION CONTAINED HEREIN, EVEN IF AMD IS EXPRESSLY ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

## **ATTRIBUTION**

© 2016 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. in the United States and/or other jurisdictions. Other names are for informational purposes only and may be trademarks of their respective owners.