



嵌入式处理器体系结构

李曦

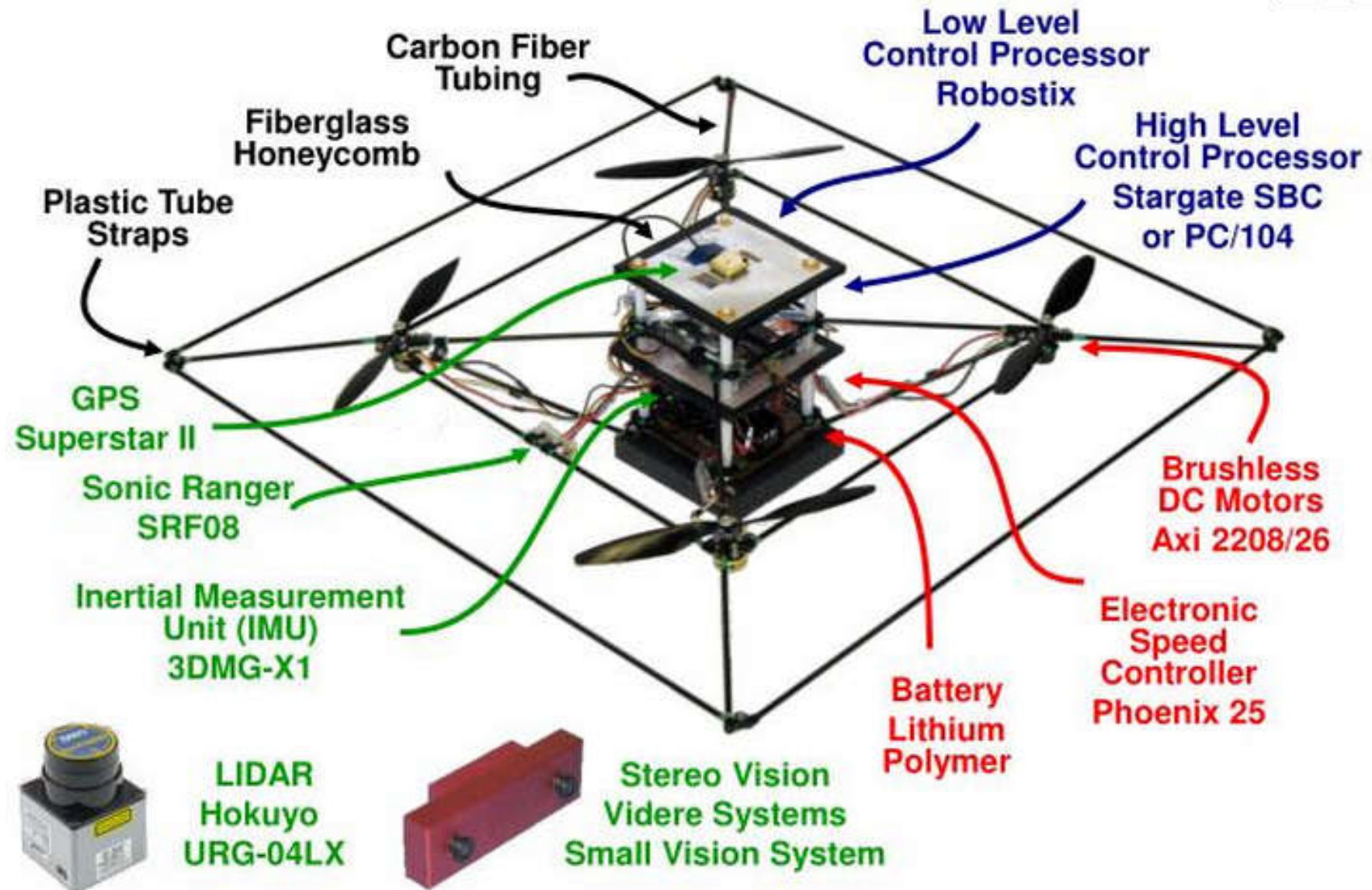
llxx@ustc.edu.cn

内容提要

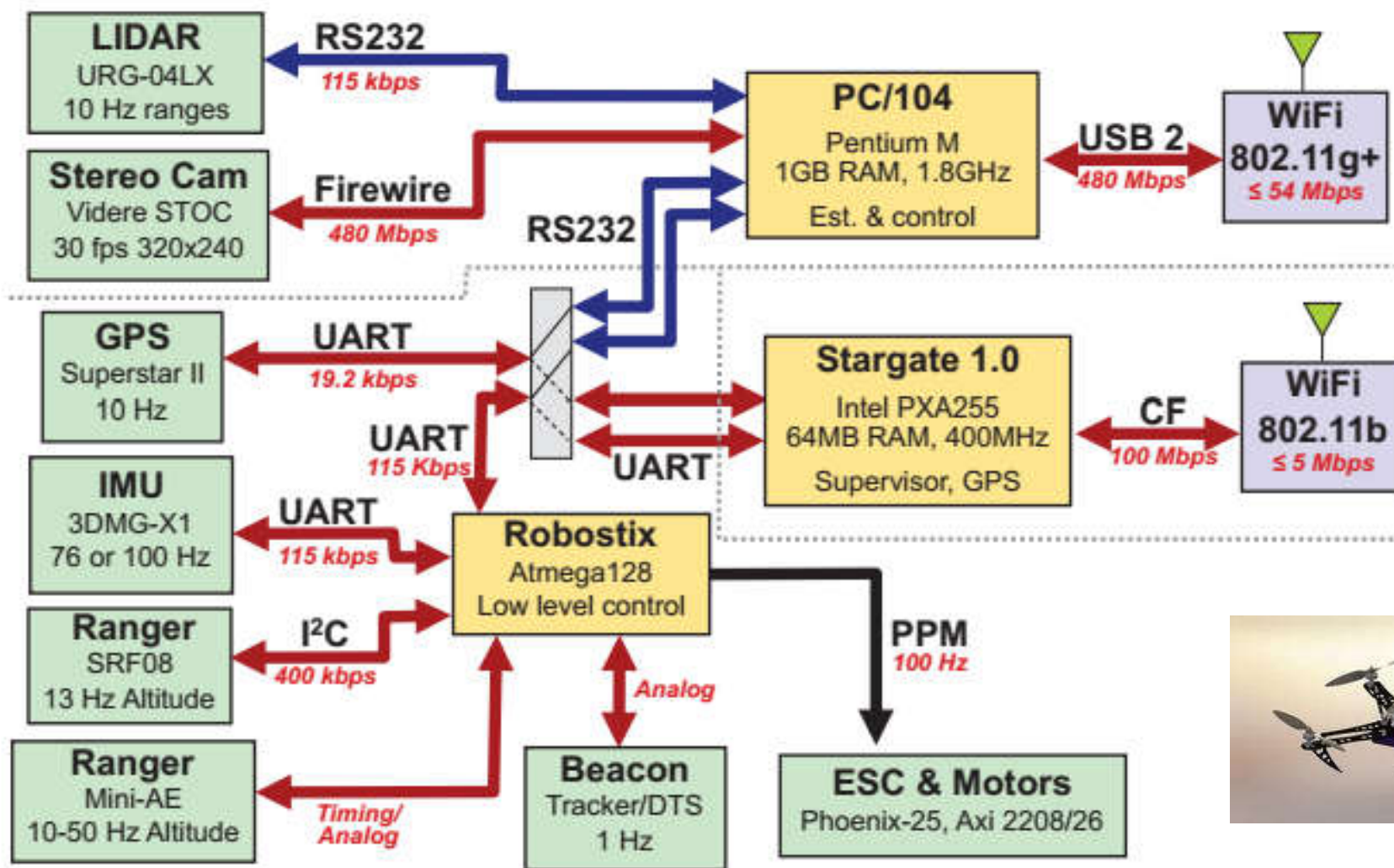
- 嵌入式处理器概述（实例）
- 通用处理器体系结构
 - CISC
 - RISC
- 嵌入式处理器
- I/O系统
 - 中断
 - DMA



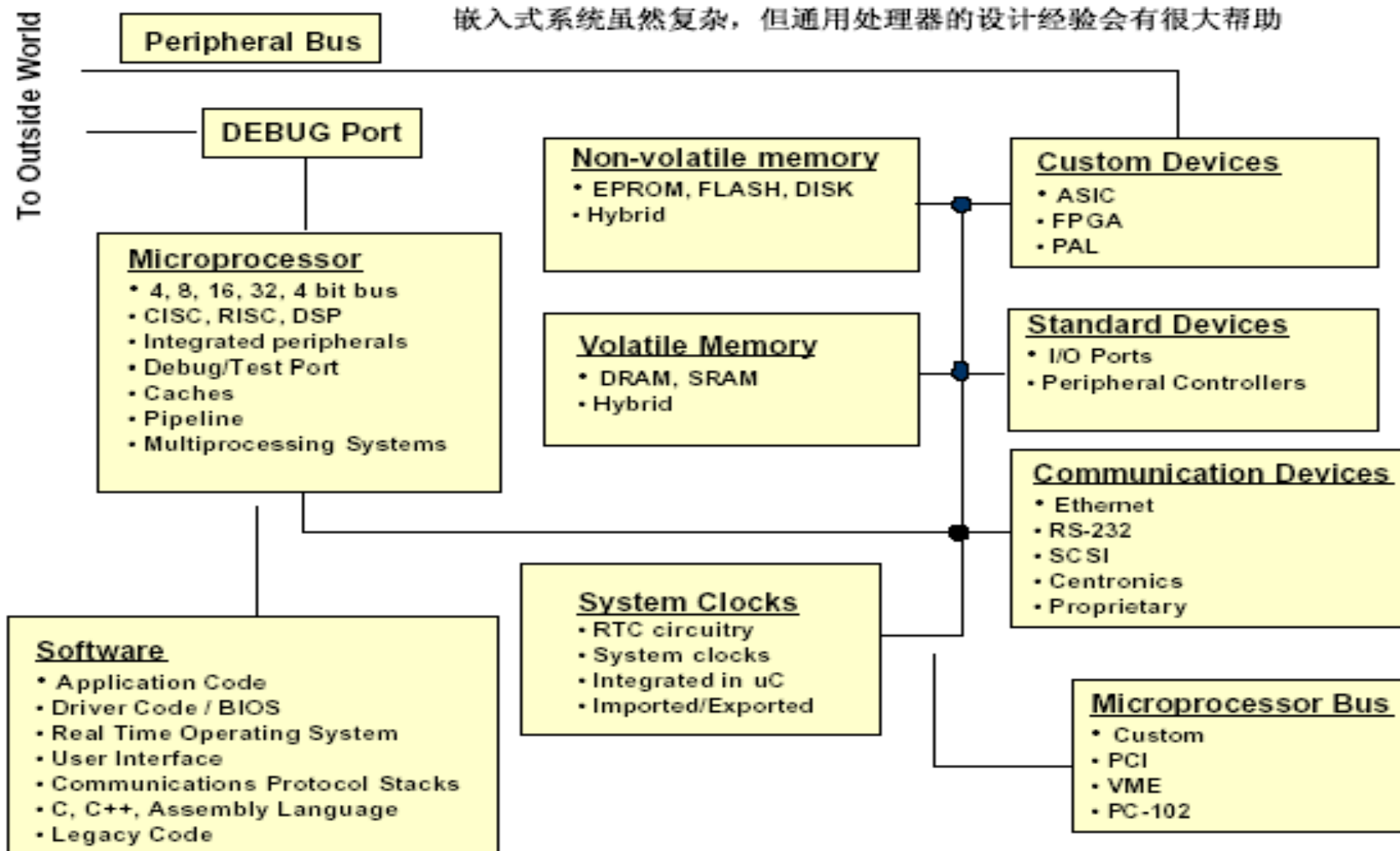
Stanford STARMAC Project



四旋翼飞控： STARMAC architecture



Typical Architecture for RTS



Microprocessor types by applications



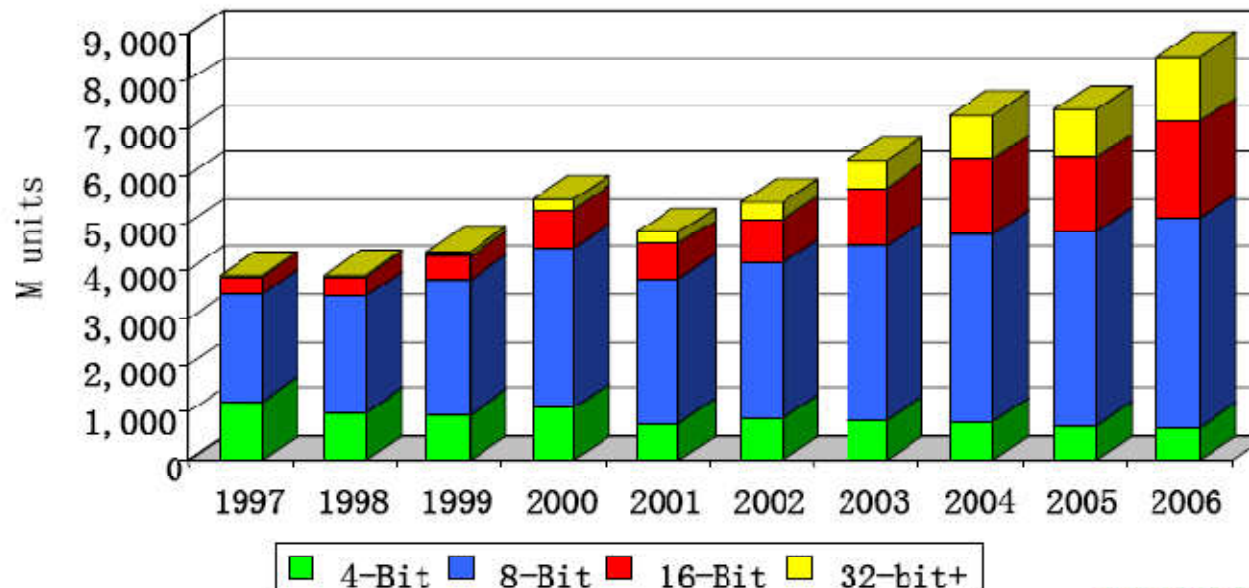
- General-purpose Microprocessor
 - Desktop applications: X86、PowerPC
 - Server applications
 - 科学和工程计算:Power1, Power2, Power3, **Power4**
 - 数据库和事务处理:RS64, RS64-II, RS64-III
- Embedded Microprocessor:
 - focused on particular application area
 - Microprocessors: Media, Graphic, Network and Communication
 - Microcontrollers
 - Digital Signal Processors (DSP)
 - System on Chips(SoC)
- Of today's microprocessors
 - **95%** go into **embedded applications**
 - **50%** of revenue stems from **embedded systems**

嵌入式微处理器

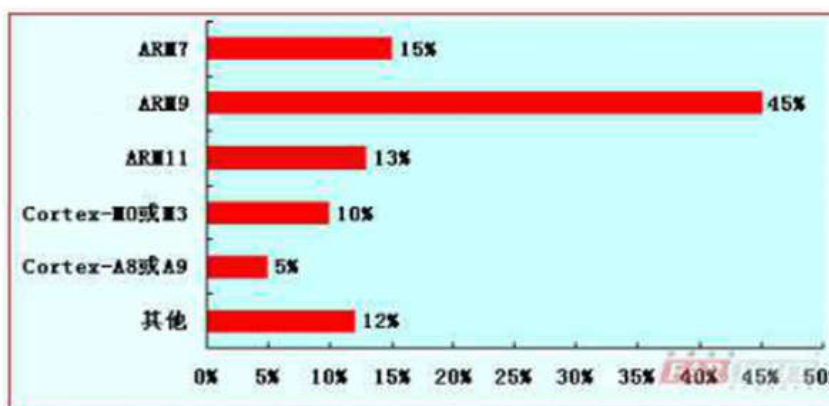
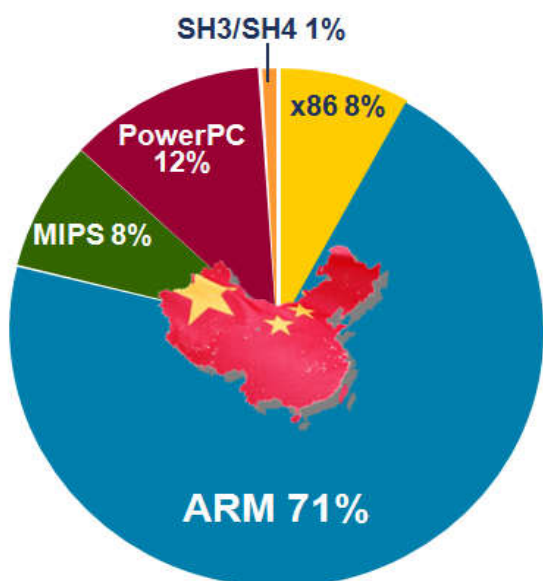


- 1 瑞萨科技
- 2 飞思卡尔半导体PPC
- 3 Microchip Technology
- 4 NEC电子
- 5 英飞凌
- 6 东芝
- 7 Atmel
- 8 三星
- 9 恩智浦半导体(NXP)
- 10 意法半导体ARM

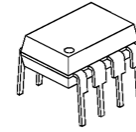
Over 60亿片 in 2003



By SEMICO



Microchip PIC单片机



- 初档8位单片机：PIC12C5XXX / 16C5X系列
 - 最早在市场上得到发展，价格较低，有较完善的开发手段，在国内应用最广；
 - PIC12C5XX：世界第一个8引脚低价位单片机
 - 512字节ROM、25字节RAM、一个8位定时器、一根输入线、5根I/O线
 - 应用：摩托车点火器。售价3~6元/片。
 - 可用于简单的智能控制等一些对单片机体积要求较高的地方，前景十分广阔。
- 中档8位单片机：PIC12C6XX/PIC16CXXX系列
 - 近年来重点发展的系列产品，品种最为丰富。
 - 增加了中断功能，指令周期达到200ns，含A / D，内部E2PROM数据存储器，双时钟工作，I2C和SPI接口，异步串行通讯（UART），模拟电压比较器、LCD驱动等。封装从8~68脚；
 - 可用于高、中、低档的电子产品设计。价格适中。
 - PIC16C74：40引脚。含：4K ROM、192字节RAM、8路A/D、3个8位定时器、2个CCP模块、三个串行口、1个并行口、11个中断源、33个I/O脚。
- 高档8位单片机：PIC17CXX系列PIC17CXX
 - 是适合高级复杂系统开发的系列产品，增加了硬件乘法器，指令周期可达成160ns，性价比最高；
 - 可用于高、中档产品的开发，如马达控制、音调合成。



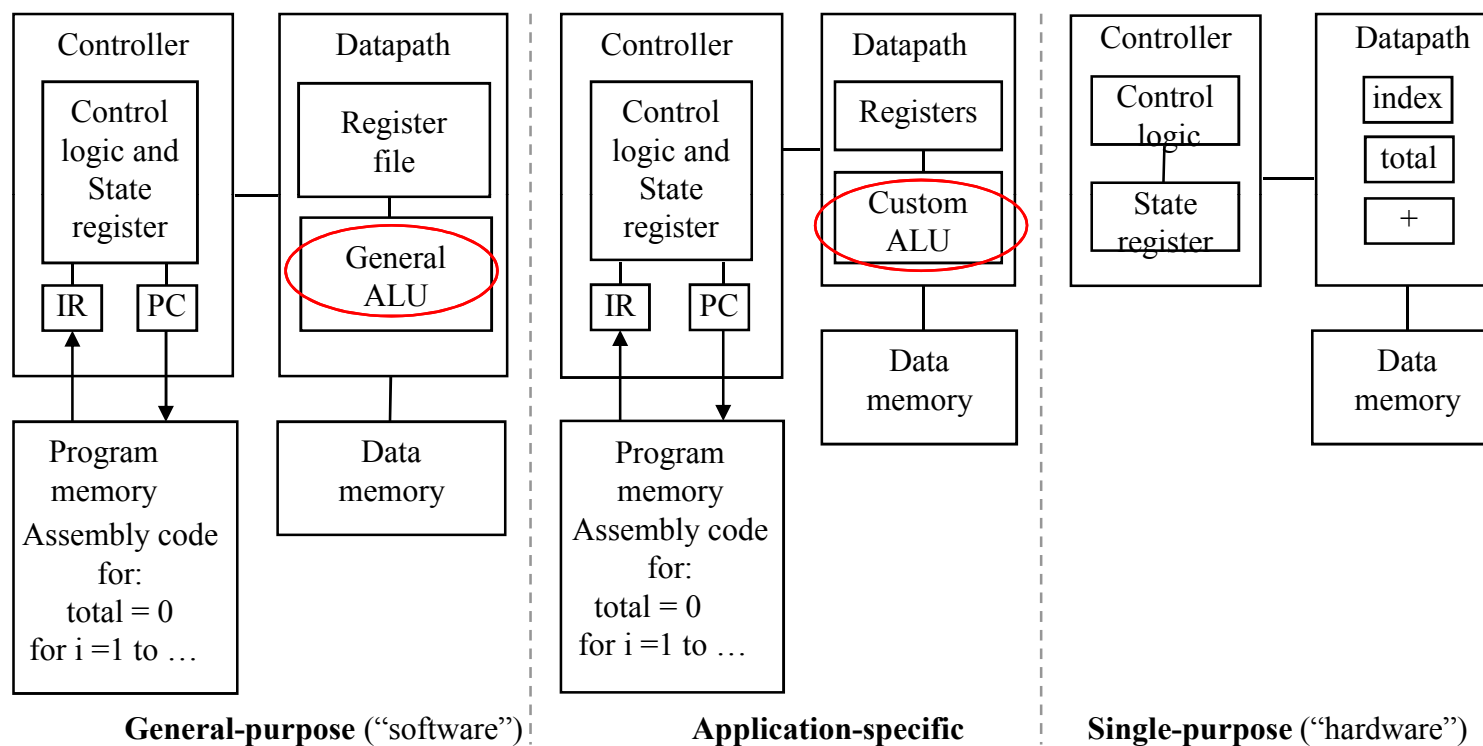
嵌入式处理器的特征

- **General rules (with exceptions):**
 1. Designed for **efficiency** (vs. ease of programming)
 2. Huge variety of processors (resulting from 1.)
 3. **Harvard** architecture
 4. **Heterogeneous** register sets
 5. Limited instruction-level parallelism or VLIW ISA
 6. Different operation **modes** (saturating arithmetic, fixed point)
 7. Specialised microcontroller & DSP instructions (bit-reversal, multiply/accumulate, bit-field addressing, modulo addressing)
 8. Multiple memory banks
 9. No “fat” (MMU, caches, memory protection, target buffers, complex pipeline logic, ...)
- **These features have to be known to the compiler!**

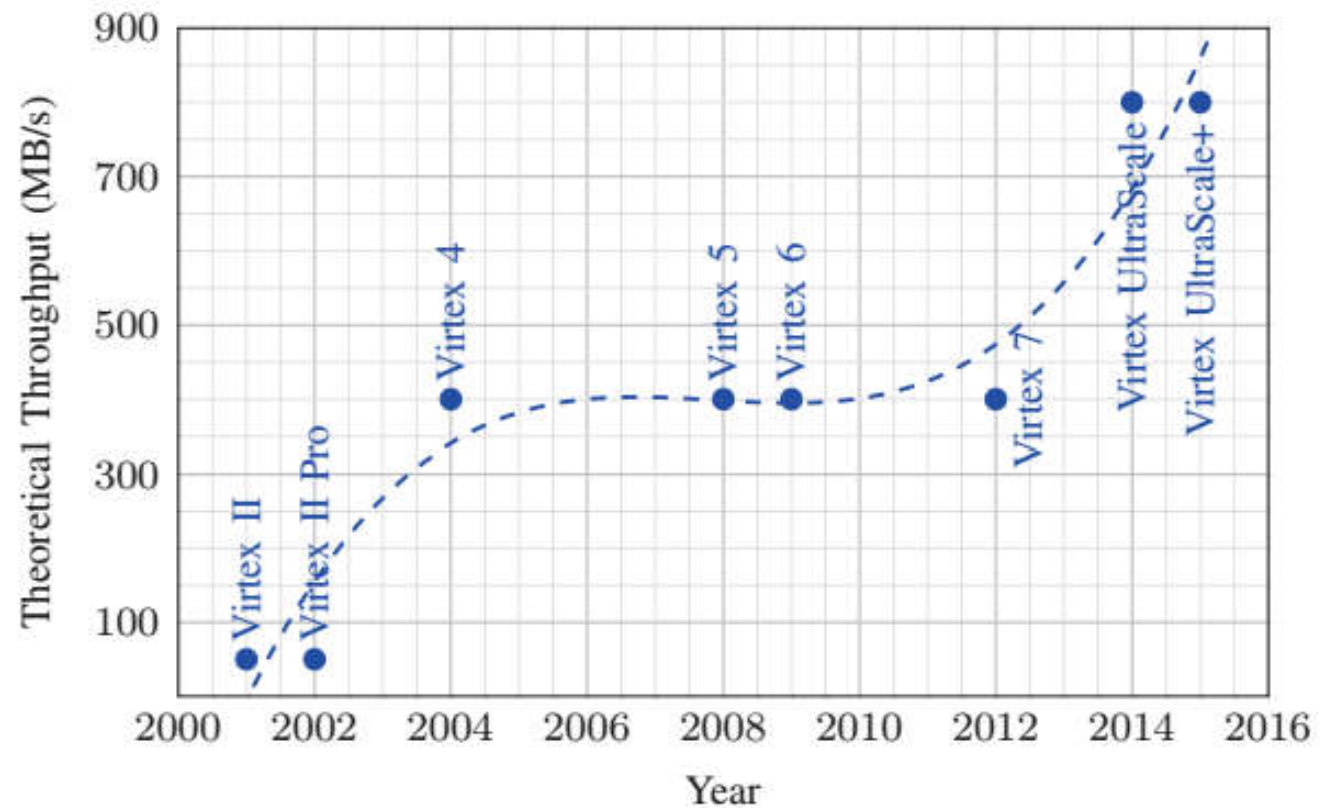
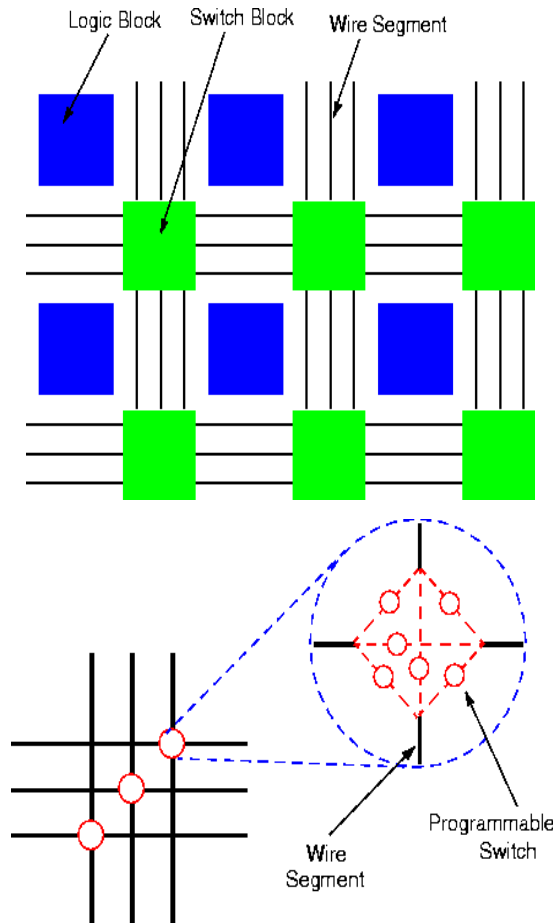
处理器技术



- “处理器”不是一定都是可编程的
 - 嵌入式微处理器并不等同于通用微处理器

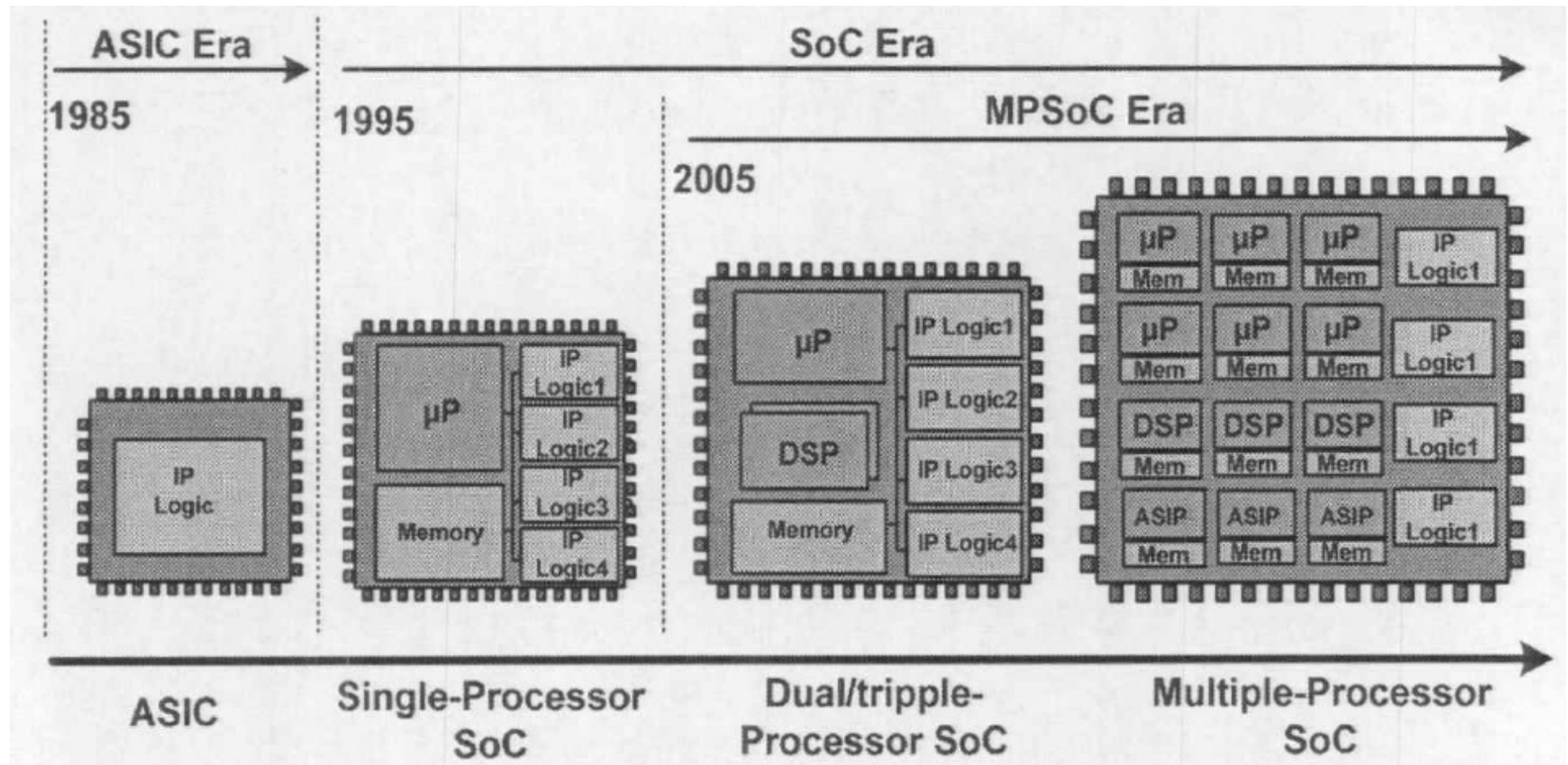


Using FPGA Field Programmable Gate Array

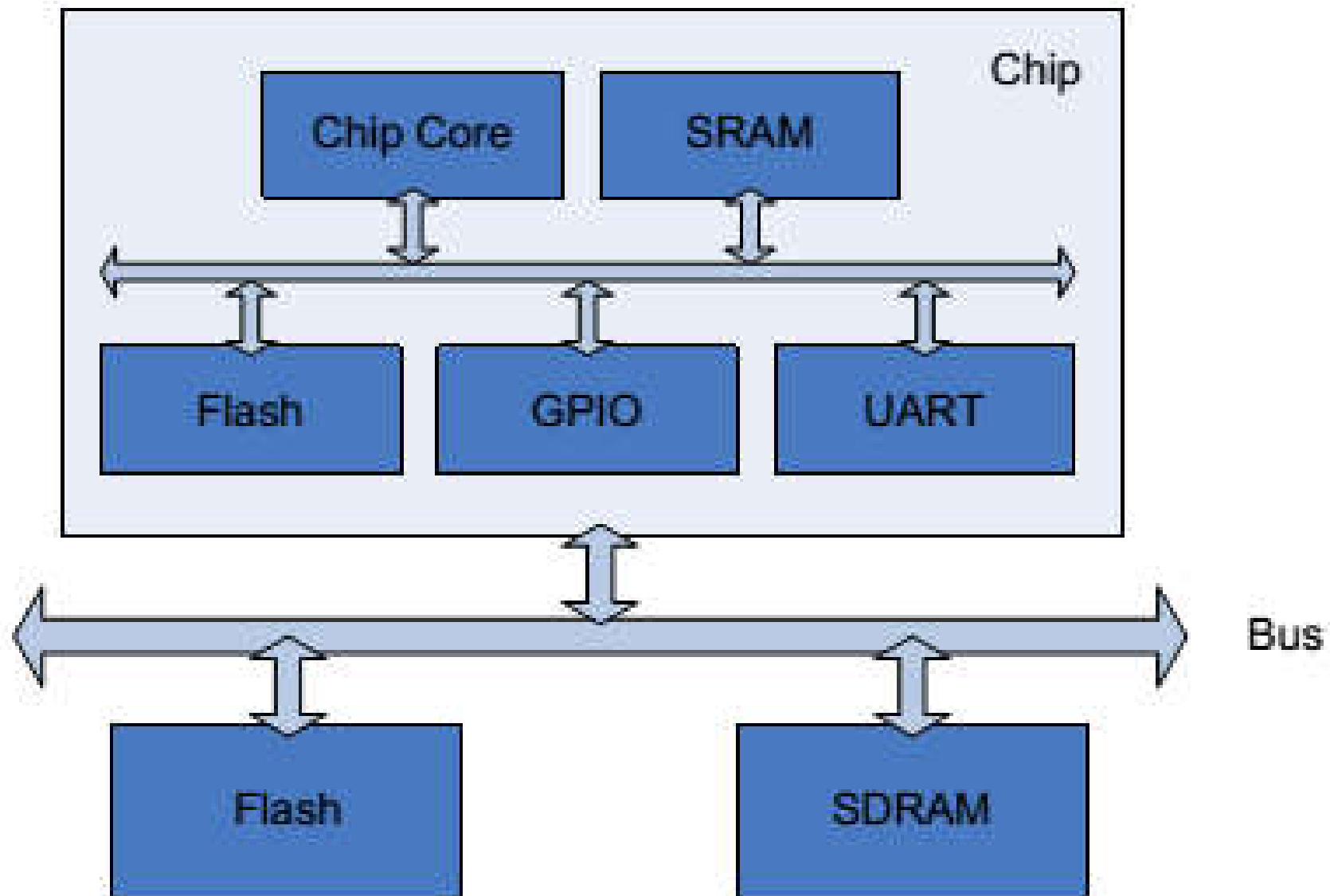


- Modern FPGA chips allow dynamic partial reconfiguration (**DPR**)
 - to reconfigure an FPGA area containing about 25% of the total resources in less than 3ms.

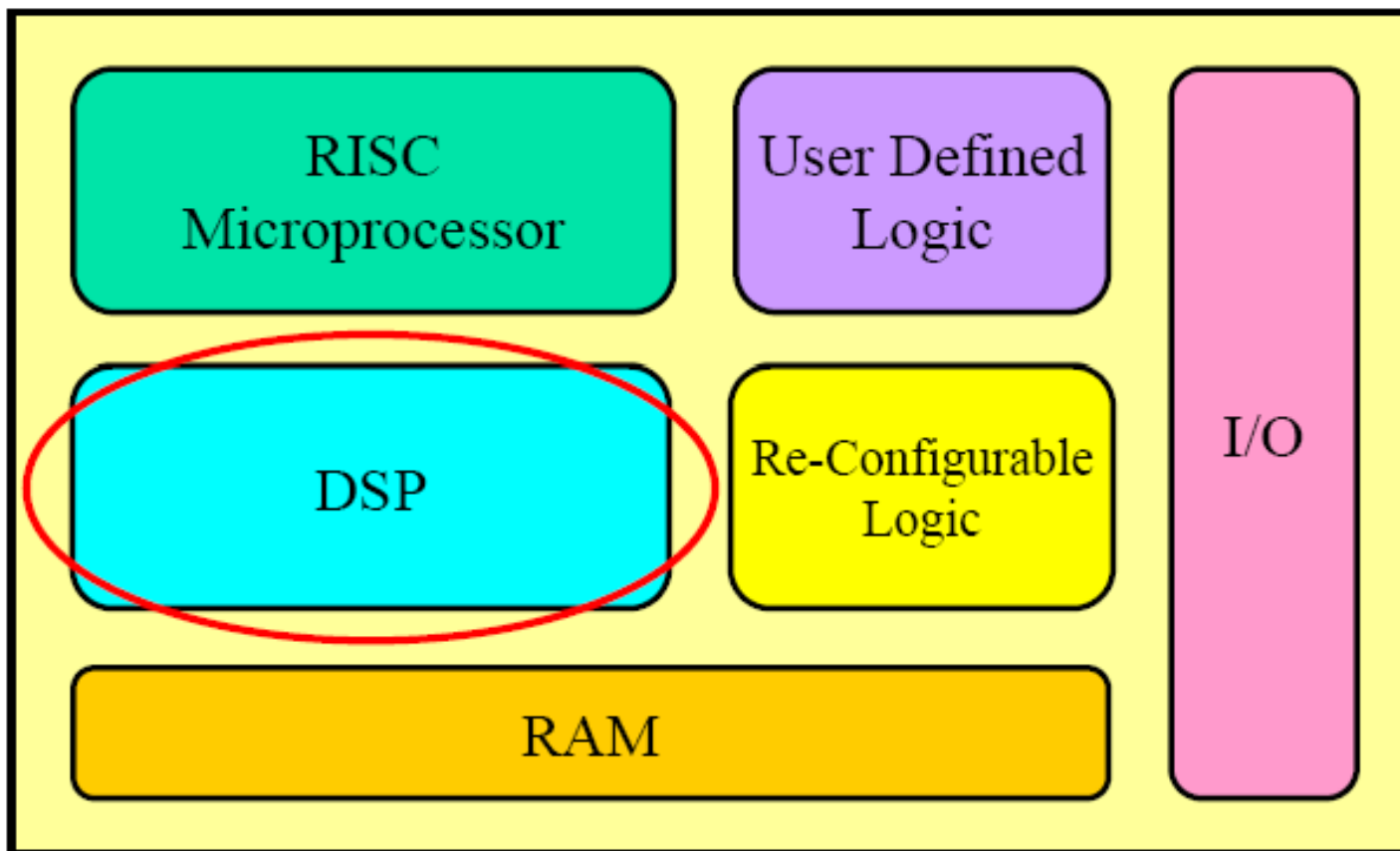
SOC Era



Architecture for SOC-based ES



SOC(System on Chip) 结构



DSP v.s. DSP !?

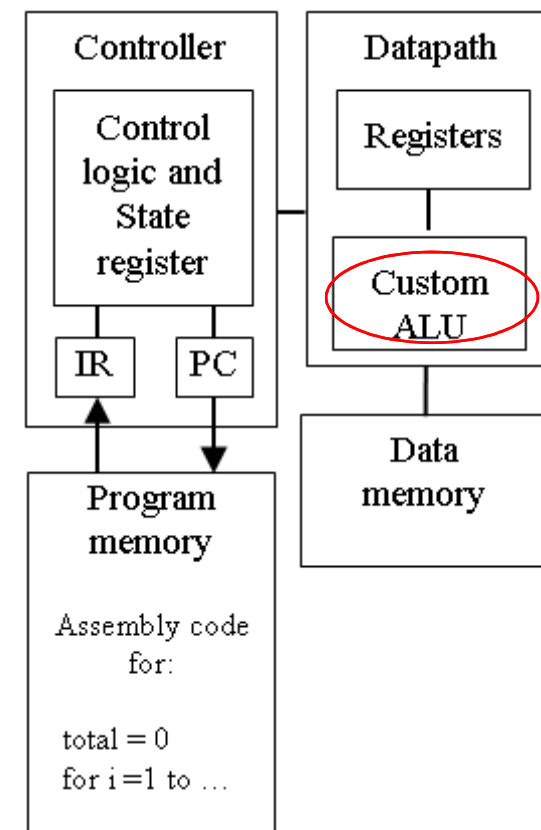
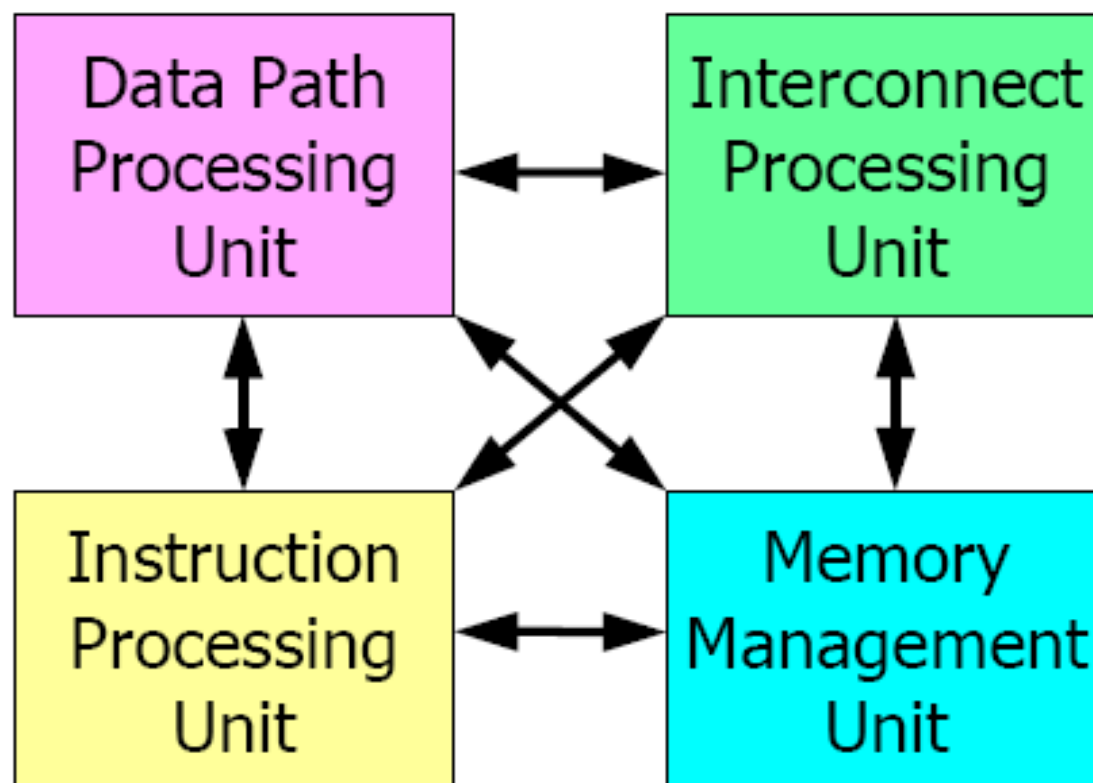


- Digital Signal Processing
 - Mathematical manipulation of digitally represented signals
- Digital Signal Processor
 - Microprocessors designed to perform Digital Signal Processing
 - Real-time performance
 - Short time-to-market
 - Upgradable solutions
 - Move Hardware Design (ASIC) to Software Design (Programming)
 - Offer the key to SOC integration

DSP结构: ASP



Processor Components





Characteristics of DSP's

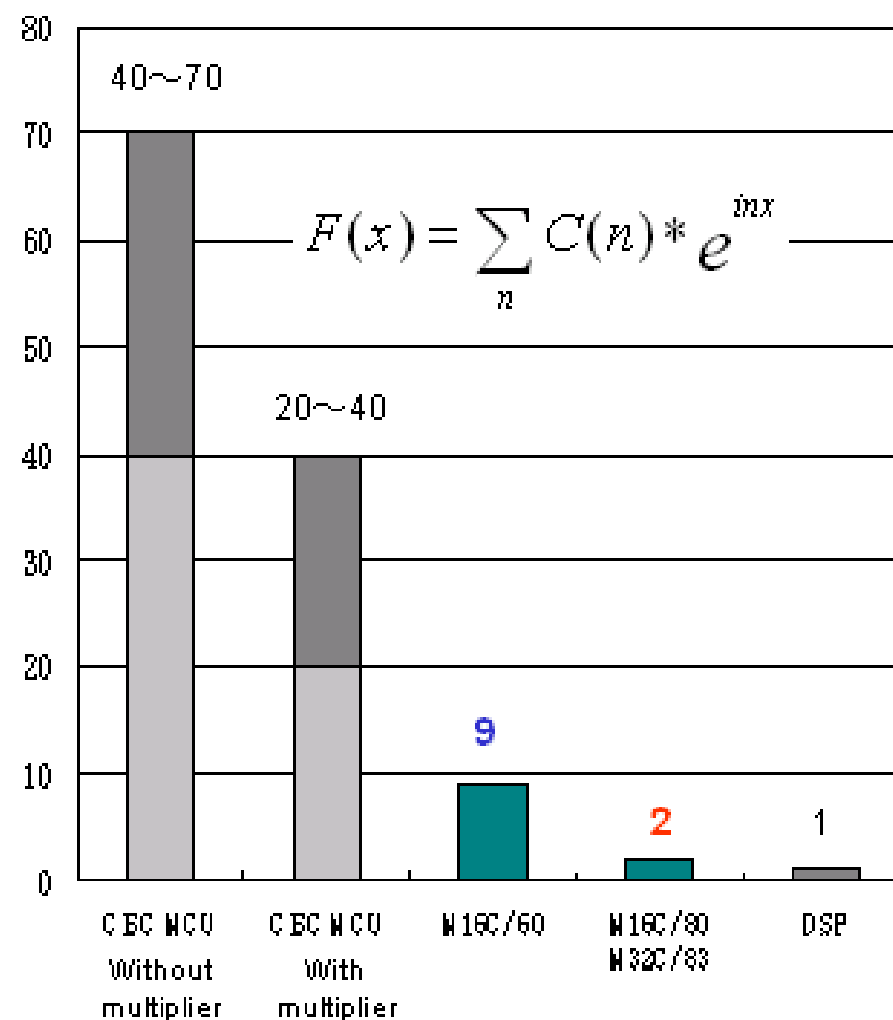
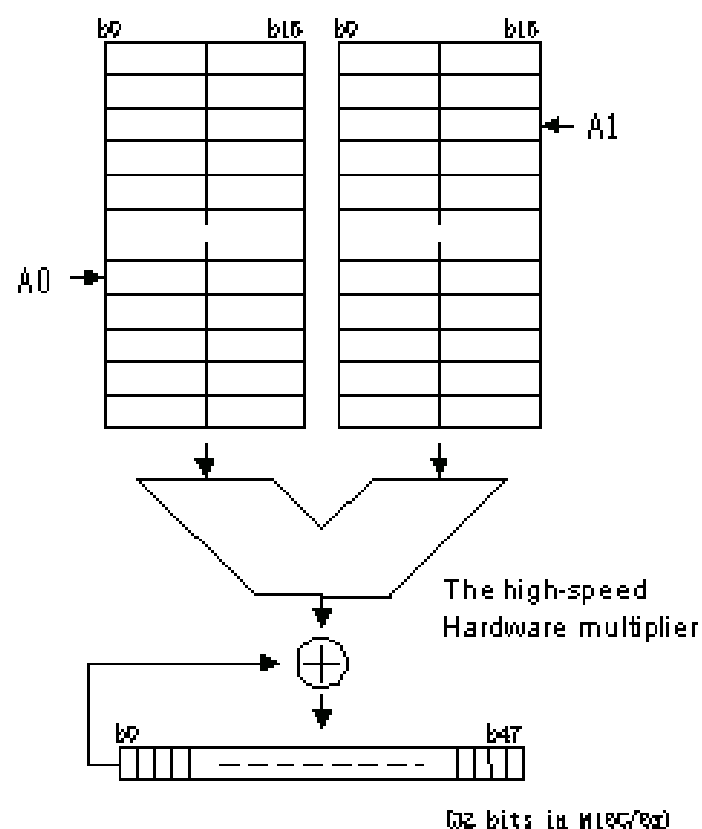
- CISC & RISC
- Harvard
 - One program fetch and multiple data R/W in each instruction cycle
- **Multiple busses** with restricted connectivity
- High I/O-rate
- Irregular register sets of different sizes
- Special purpose registers
- Special registers for zero-overhead loops
- **Multiply-add-accumulate** in one instruction
 - Single cycle multiply-accumulate (MAC)
- Fixed-point operations
- Hardware supported addressing modes
 - Register indirect addressing with post-increment
 - Bit-reversal addressing for FFT
 - Modulo addressing for hardware circular buffers



DSP的特殊指令

Sum-of-products operation for one data is executed in two cycles.

General multiply processing also is executed in three cycles using a high-speed hardware multiplier.

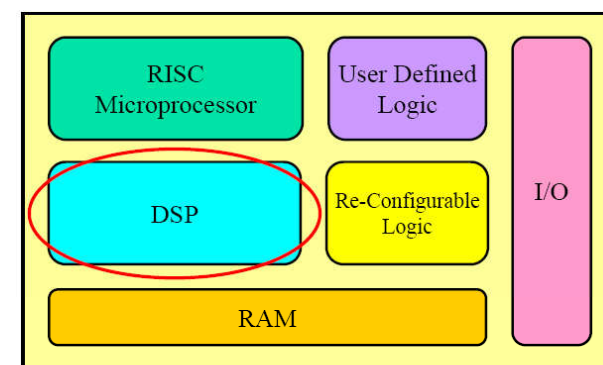




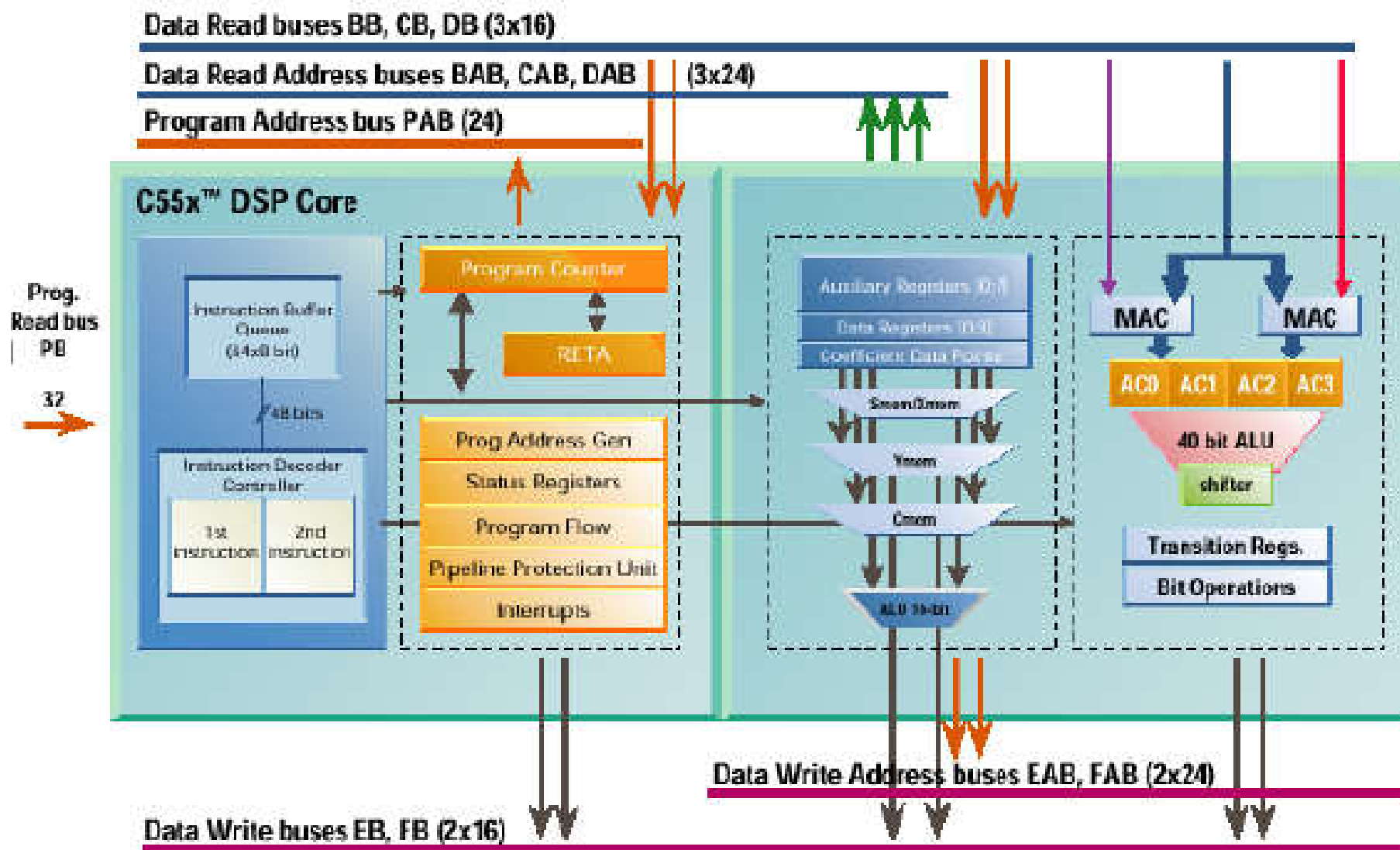
DSP的低功耗策略

$$P = P_{dyn} + P_{sc} + P_{lk} = 0.5C_L V_{DD}^2 A f + I_{sc} V_{DD} A + I_{lk} V_{DD}$$

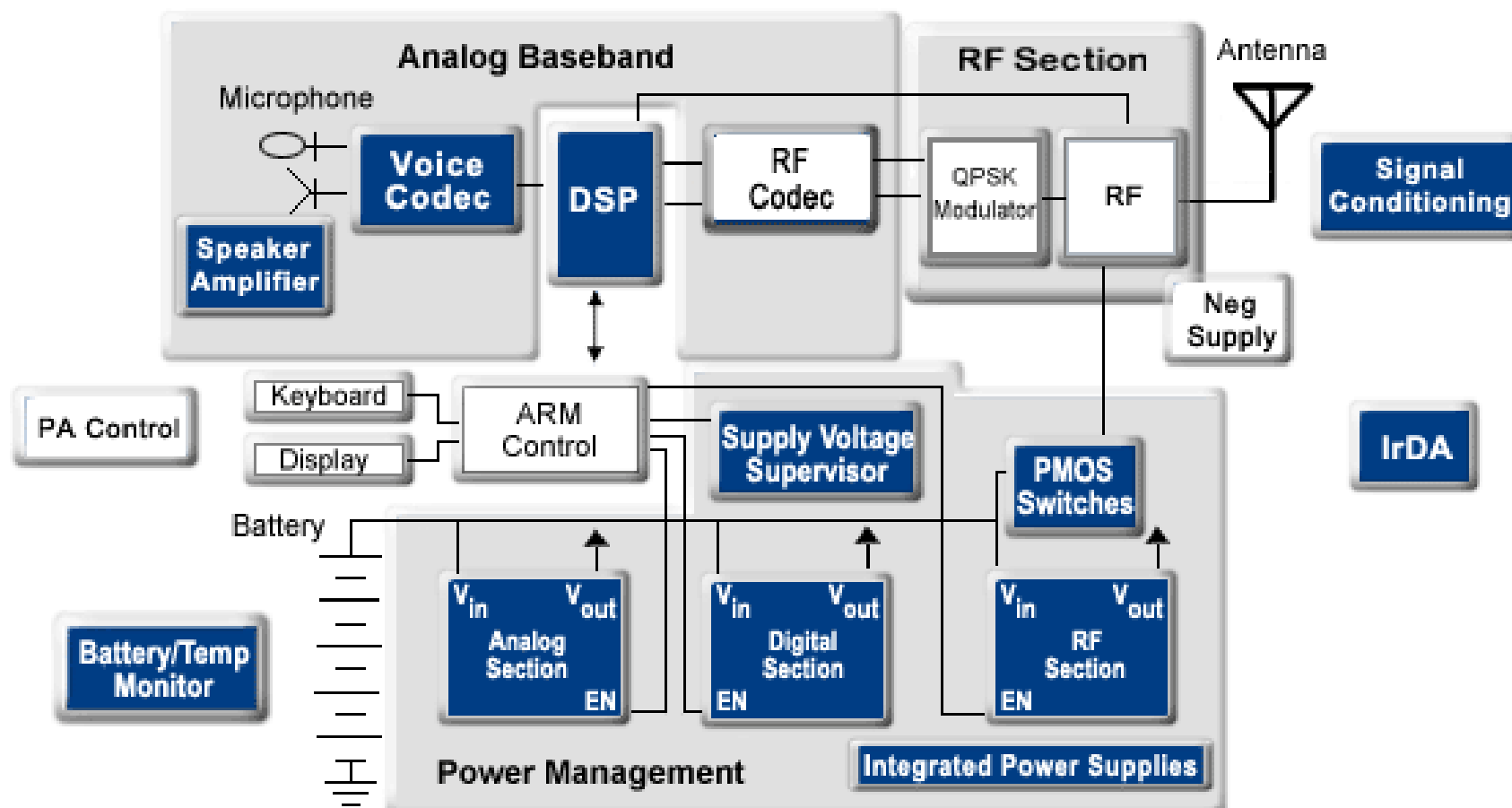
- Power Management
 - Gated clock
 - Power off un-used parts
 - Reduce clock rate of idle parts
- Variable Instruction Length
- Compressed Instruction
- Full-Custom Data Path Layout
- Technology and Voltage Scaling



TI TMS320C55x: 多总线



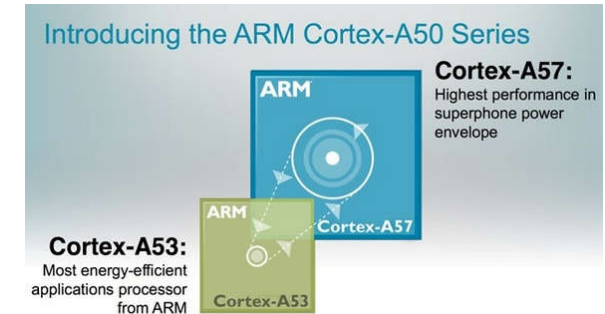
基于DSP的Cellphone



MPSoCs



- Homogeneous MPSoCs
- Heterogeneous MPSoCs
 - *asymmetric multi-core* architecture
 - *single-ISA heterogeneous* architecture
- ARM big.LITTLE 处理器
 - Cortex-A15: 1.8GHz主频的四核MPCore, 高性能
 - Cortex-A7: 1.2GHz主频的四核MPCore, 节能
 - 同一应用程序软件在二者之间无缝切换
 - 为每个任务选择最佳处理器, 可以使电池的寿命延长70%。

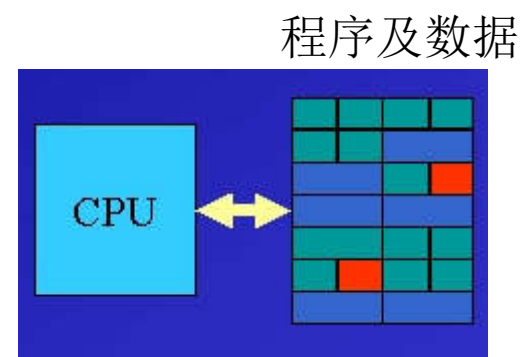


计算机系统结构



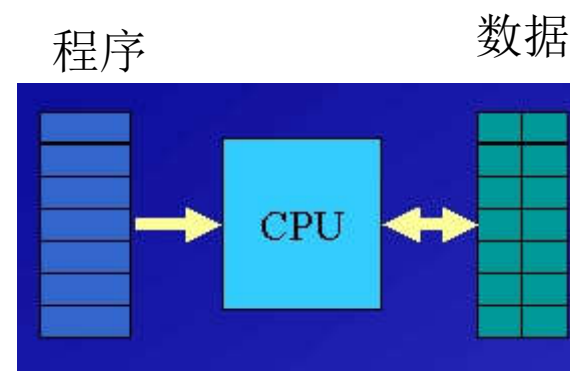
- 冯. 诺依曼结构 (von Neumann arch)

- 系统组成: ALU, Controller, Memory, Input, Output
- 程序指令和数据放在同一存储器的不同物理位置, CPU通过同一条总线访问程序和数据, 程序指令和数据的宽度是相同的
- 存储程序(stored program): 程序以数字形式存在, 可以与数据一样被读写



- 哈佛体系结构 (Harvard architecture)

- 程序与数据有单独的存储器, 它们通过不同的总线来访问
 - 首先到程序指令存储器中读取程序指令, 解码后得到数据地址, 再到相应的数据存储器中读取数据, 执行指令。
 - 在执行一条指令的同时可以预先读取下一条指令
- 指令和数据可以有不同的数据宽度
 - 如PIC的程序指令是14位, 而数据是8位



处理器ISA



- CISC

- Intel推出MCS-48、MCS-51(1980)、MCS-96系列
 - 20世纪80年代中期以后，Intel以专利转让的形式把8051内核给了许多半导体厂家
 - 如AMTEL、PHILIPS、ANALOG DEVICES、DALLAS等。
 - 这些厂家生产的芯片是MCS-51系列的兼容产品，准确地说是与MCS-51指令系统兼容的单片机。
- 应用：控制关系较复杂的场合，如通讯产品、工业控制系统

- RISC

- Microchip的PIC系列、Zilog的Z86系列、Atmel的AT90S系列、韩国三星公司的KS57C系列4位单片机
- 应用：控制关系较简单的小家电

- VLIW

- TI DSP处理器

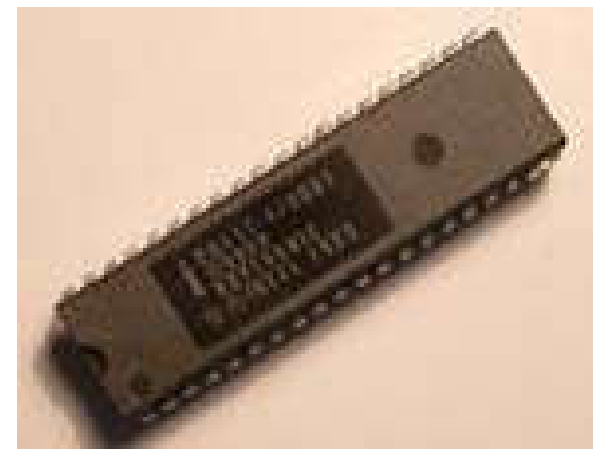
复杂指令集计算机CISC



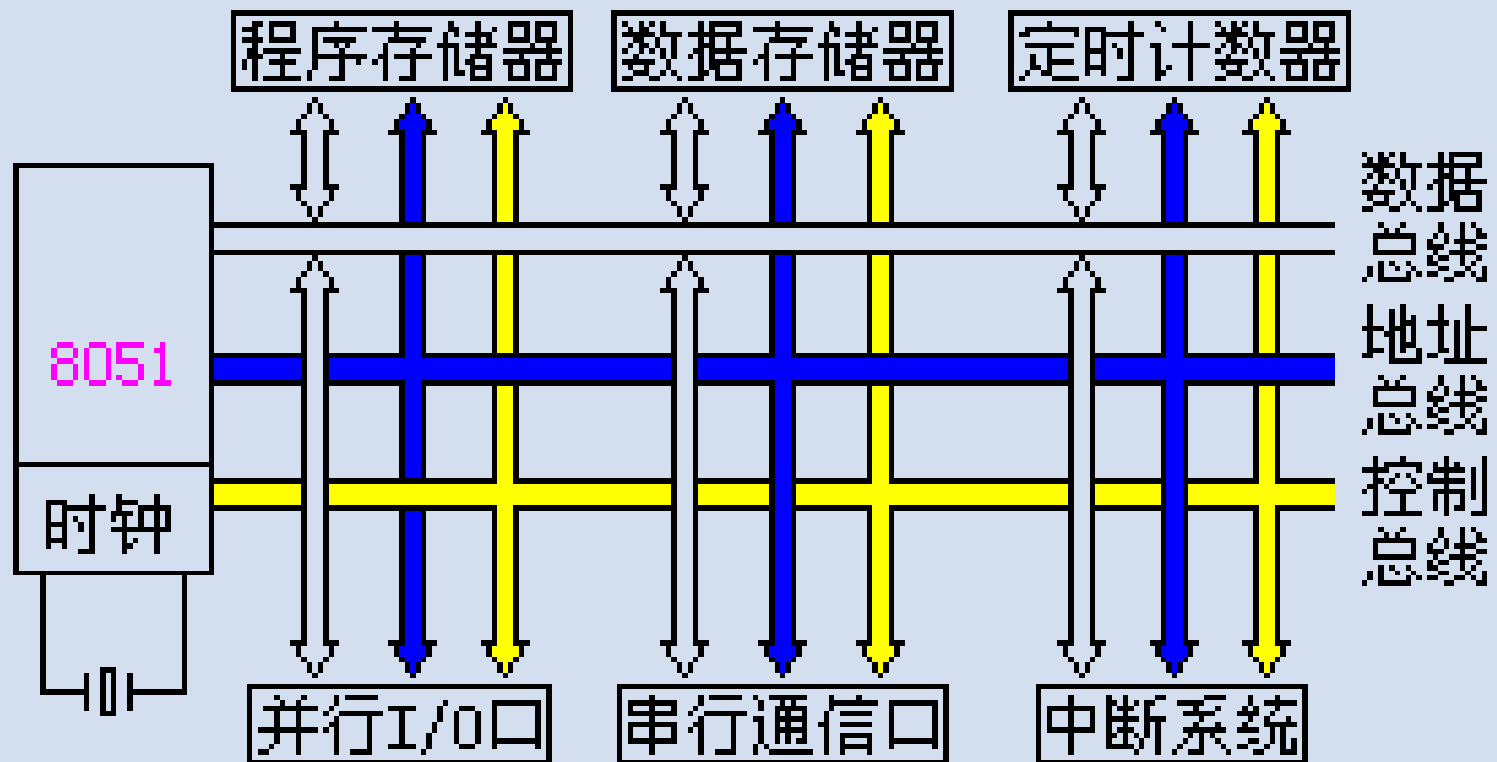
- 背景：
 - 存储资源紧缺，设置一些功能复杂的指令以减少完成一个任务所需的指令数目
 - 增强指令功能，通过减少程序中指令数达到提高运行速度的目的
- 特点：
 - 指令格式不固定，指令可长可短，操作数可多可少
 - 寻址方式复杂多样，操作数可来自寄存器，也可来自存储器
 - 使用微代码。指令集存储在控存里（比主存的速度快很多）
 - 允许设计师实现CISC体系机器的向上相容
 - 新的系统可以使用一个包含早期系统的指令超集
 - 微程序指令的格式与高阶语言相匹配，因而编译器的设计较简单
 - $CPI > 5$ ，指令越复杂，CPI越大
 - 中断响应时间难于预期，实时性不好

8051单片机

- 8位CPU
 - 111条指令，大部分为单字节指令
 - 21个专用寄存器
 - 逻辑操作位寻址功能
 - 4kbytes 程序存储器(ROM)
 - 128bytes的数据存储器(RAM)
 - 5个中断源，2个优先级
 - 两个外中断、两个定时/计数器中断和一个串行中断
 - 2个可编程定时/计数器（16位）
 - 32条I/O口线（4*8bits并口(P0、P1、P2或P3)）
 - 一个全双工串行通信口
- 外部ROM：程序存储器寻址空间64kB
- 外部RAM：数据存储器寻址空间64kB
- 双列直插40PinDIP封装
- 单一+5V电源供电
- 工作频率为2~12MHz



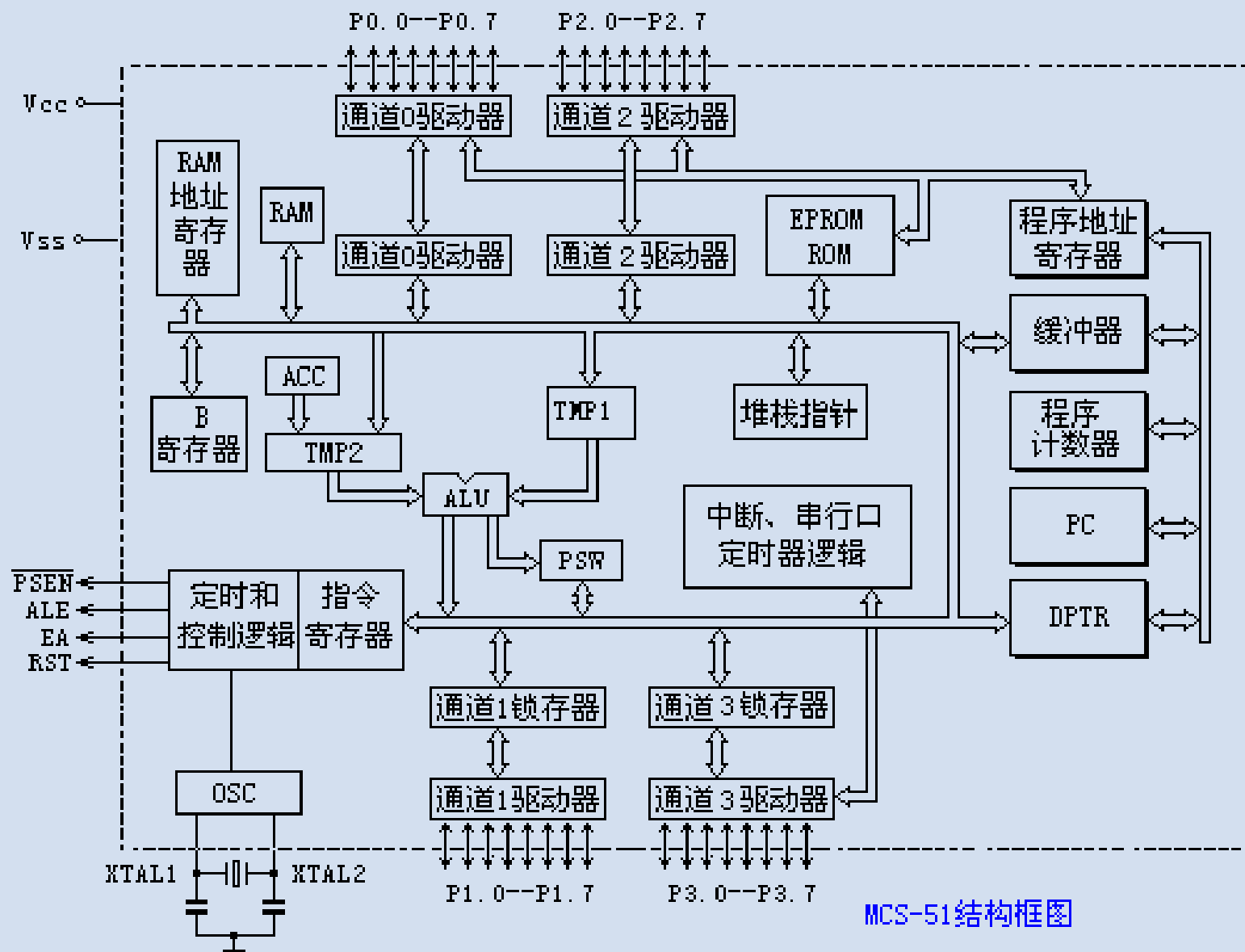
8051 内部结构



www.zymcu.com

- 普林斯顿(Princeton)结构——冯.诺伊曼结构
 - 程序存储器与数据存储器合二为一
- 哈佛(Harvard)结构——8051
 - 程序存储器和数据存储器分开

MCS-51系列单片机的内部结构

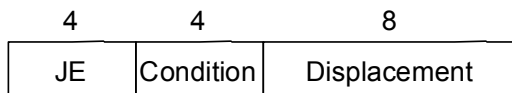


MCS-51结构框图

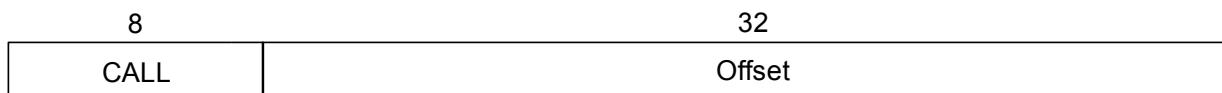
X86指令格式



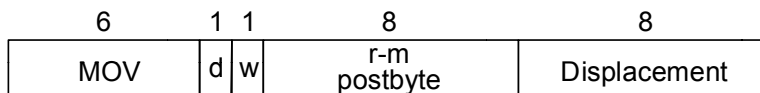
a. JE EIP + displacement



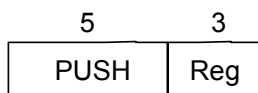
b. CALL



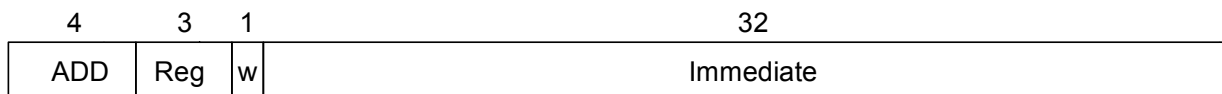
c. MOV EBX, [EDI + 45]



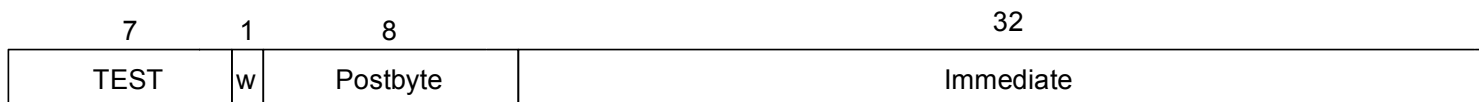
d. PUSH ESI



e. ADD EAX, #6765



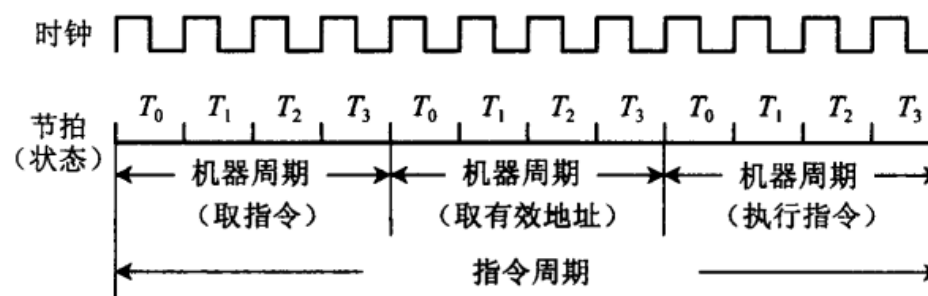
f. TEST EDX, #42



Example (8051)

007B:	0581	INC SP
007D:	18	DEC R0
007E:	EB	MOV A,R3
007F:	C6	XCH A,@R0
0080:	C9	XCH A,R1
0081:	08	INC R0
0082:	CA	XCH A,R2
0083:	C6	XCH A,@R0
0084:	CA	XCH A,R2
0085:	08	INC R0
0086:	F6	MOV @R0,A
0087:	7BFF	MOV R3,#FF
0089:	11A7	ACALL 00A7
008B:	D0E0	POP ACC

t	dt	PC	Source
-15	2	033A	MOV SCON,#52
-13	2	033D	MOV TMOD,#20
-11	2	0340	MOV TCON,#69
-9	2	0343	MOV TH1,#F3
-7	1	0346	MOV R3,#05
-6	1	0348	MOV R2,#03
-5	1	034A	MOV R1,#1F
-4	2	034C	LCALL ?PRINTF1
-2	2	0079	MOV R0,SP
0	0	007B	INC SP





CISC的缺陷

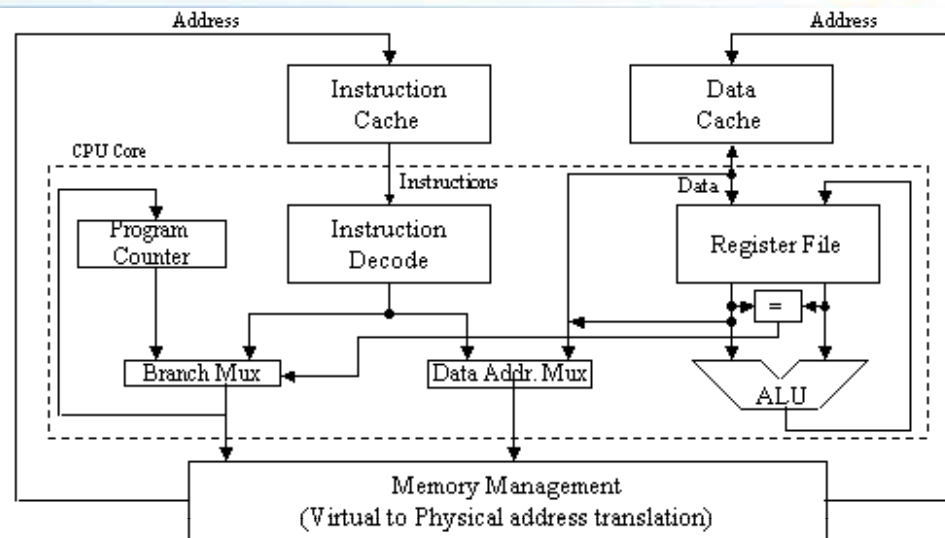
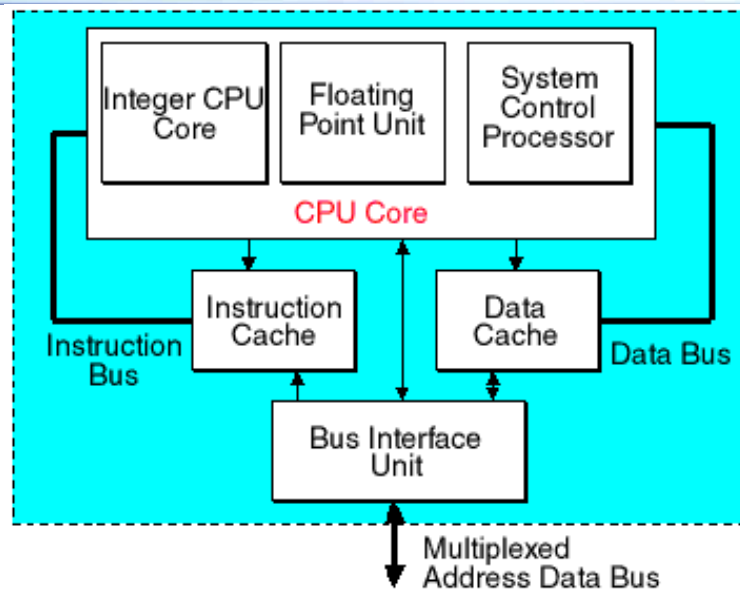
- 指令使用频度不均衡
 - “80~20” 理论
 - 复杂指令往往是低频度指令
- 大量复杂指令的控制逻辑不规整，不适于VLSI工艺
 - 微程序制约速度提高(微码的存储速度比CPU慢5-10倍)
- CISC指令的格式长短不一，需要不同的时钟周期来完成。
 - 执行较慢的指令影响系统性能。
 - 不利于采用指令级并行技术
- 软硬功能分配
 - 可划分的粒度变大了，灵活性小

典型指令使用频度

指令类型	使用指令使用频度
数据传送类	43%
转/跳控制类	23%
算术运算类	15%
比较类	13%
逻辑运算类	5%
其他	1%



RISC基本设计思想

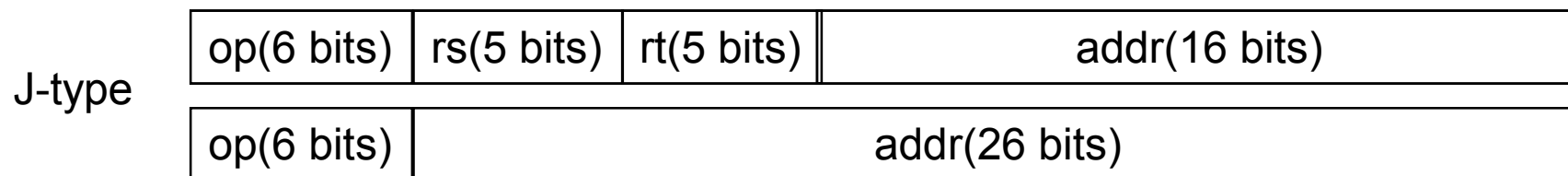
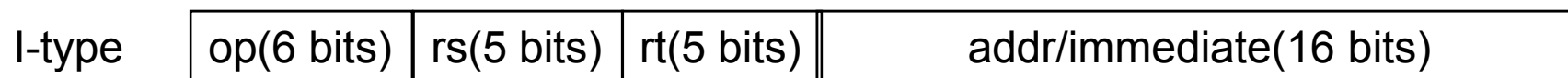
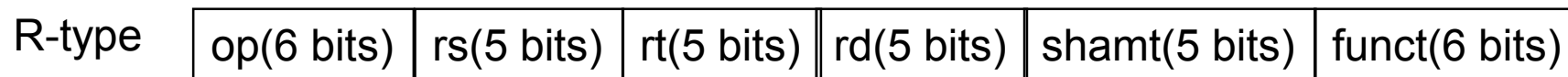


- 精简指令集：保留最基本指令(运算指令、加载、存储指令和转移指令)，减小CPI：
 $CPUtime = IC * CPI * CC$ IC—程序中指令数 CPI—每条指令执行周期数 CC—时钟周期
- 复杂指令可通过对简单基本的指令组合而成
- 每条指令的长度相同，大部分指令可以在一个机器周期里完成，CPI=1
- 采用多级指令流水线结构, 处理器在同一时间内可执行多条指令，IPC>1
- 采用加载(Load)/存储(Store)结构，统一存储器访问方式，只允许Load和Store指令执行存储器操作，其余指令均对寄存器操作。
- 大大增加通用寄存器的数量，ALU只与寄存器文件直接连接。
- 采用高速缓存(cache)结构，并基于哈佛结构



MIPS指令格式

- 100余条指令，32个通用寄存器
- 指令格式：定长32位
 - R-type: arithmetic instruction
 - I-type: data transfer
 - J-type: branch instruction(conditional & unconditional)





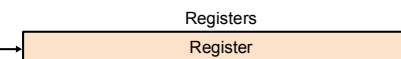
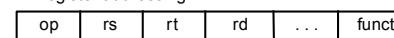
MIPS寻址模式

- 寄存器寻址: R-type
- 基址寻址: I-type
- 立即寻址
- 相对寻址
- 伪直接寻址
(pseudodirect addressing)
 - 26位形式地址左移2位, 与PC的高4位拼接

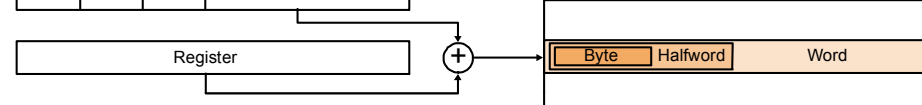
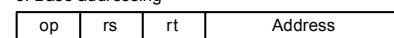
1. Immediate addressing



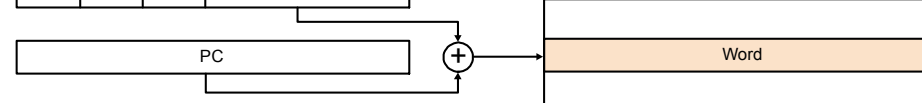
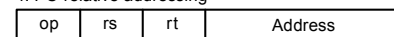
2. Register addressing



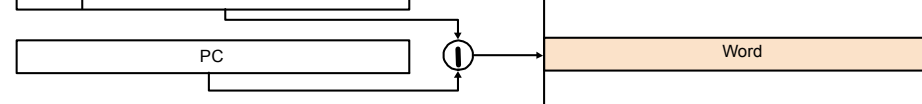
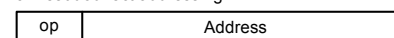
3. Base addressing



4. PC-relative addressing



5. Pseudodirect addressing

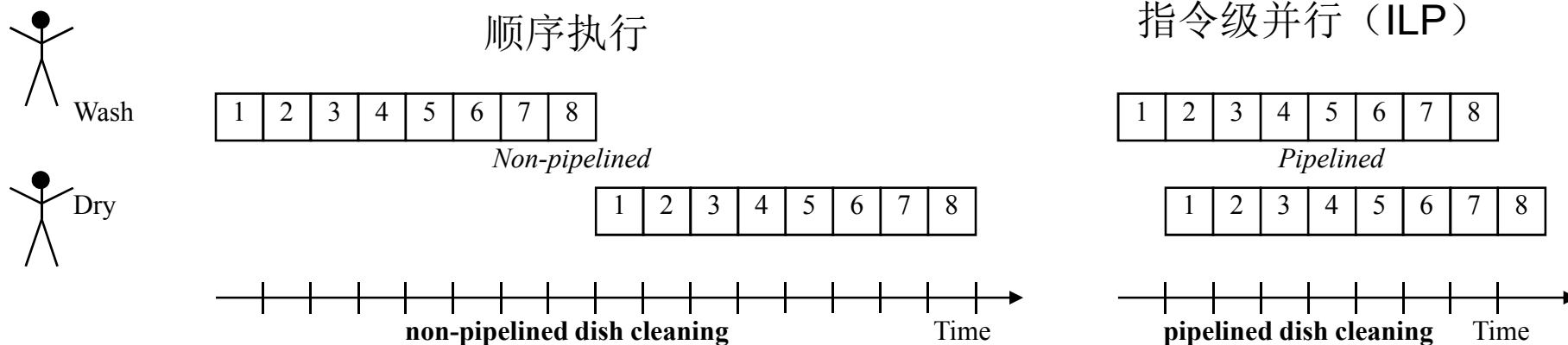


RISC CPU Example (ARM)



0x00008110	mov	r6,#1
0x00008114	add	r0,r6,r6,ls1 #1
0x00008118	add	r6,r0,#1
0x0000811c	cmp	r6,r10
0x00008120	ble	0x8114 ; (shell_sort + 0x14)
0x00008124	mov	r1,r6
0x00008128	mov	r0,#3
0x0000812c	bl	__rt_sdiv
0x00008130	mov	r6,r0
0x00008134	add	r8,r6,#1
0x00008138	b	0x817c ; (shell_sort + 0x7c)
0x0000813c	ldr	r9,[r5,r8,ls1 #2]
0x00008140	mov	r4,r8
0x00008144	b	0x8154 ; (shell_sort + 0x54)
0x00008148	ldr	r0,[r5,r7,ls1 #2]

流水线技术

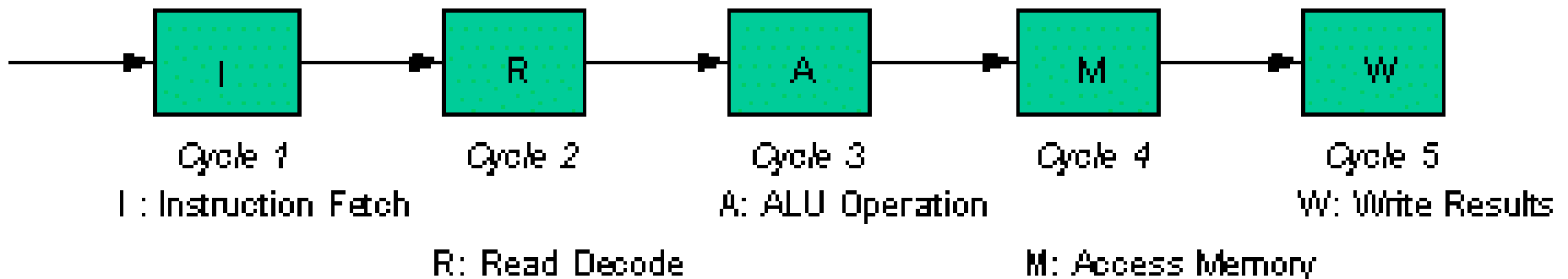


- 流水过程由多个相互联系的子过程组成，每个子过程称为流水线的“级”或“段”。
 - 一条流水线的段数被称为流水线的深度，或“流水深度”。
 - 段寄存器：暂存中间结果
- 各个功能段所需时间尽量相等
 - 否则，消耗时间长的功能段将成为流水线的瓶颈，会造成流水线的阻塞或断流。
 - 每个功能段所需的时间一般为一个机器周期（cycle）。

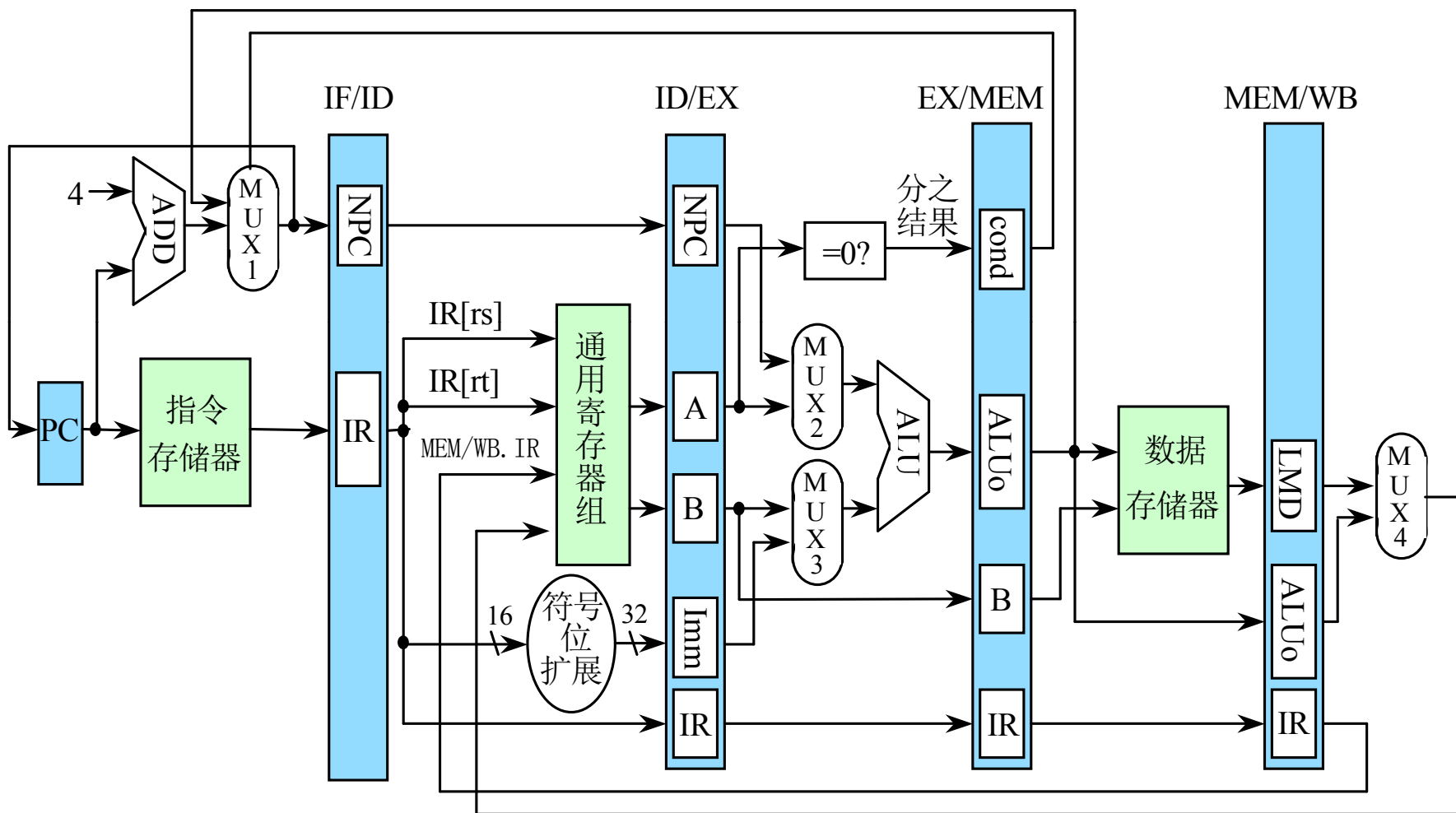


Pipelining of Instructions

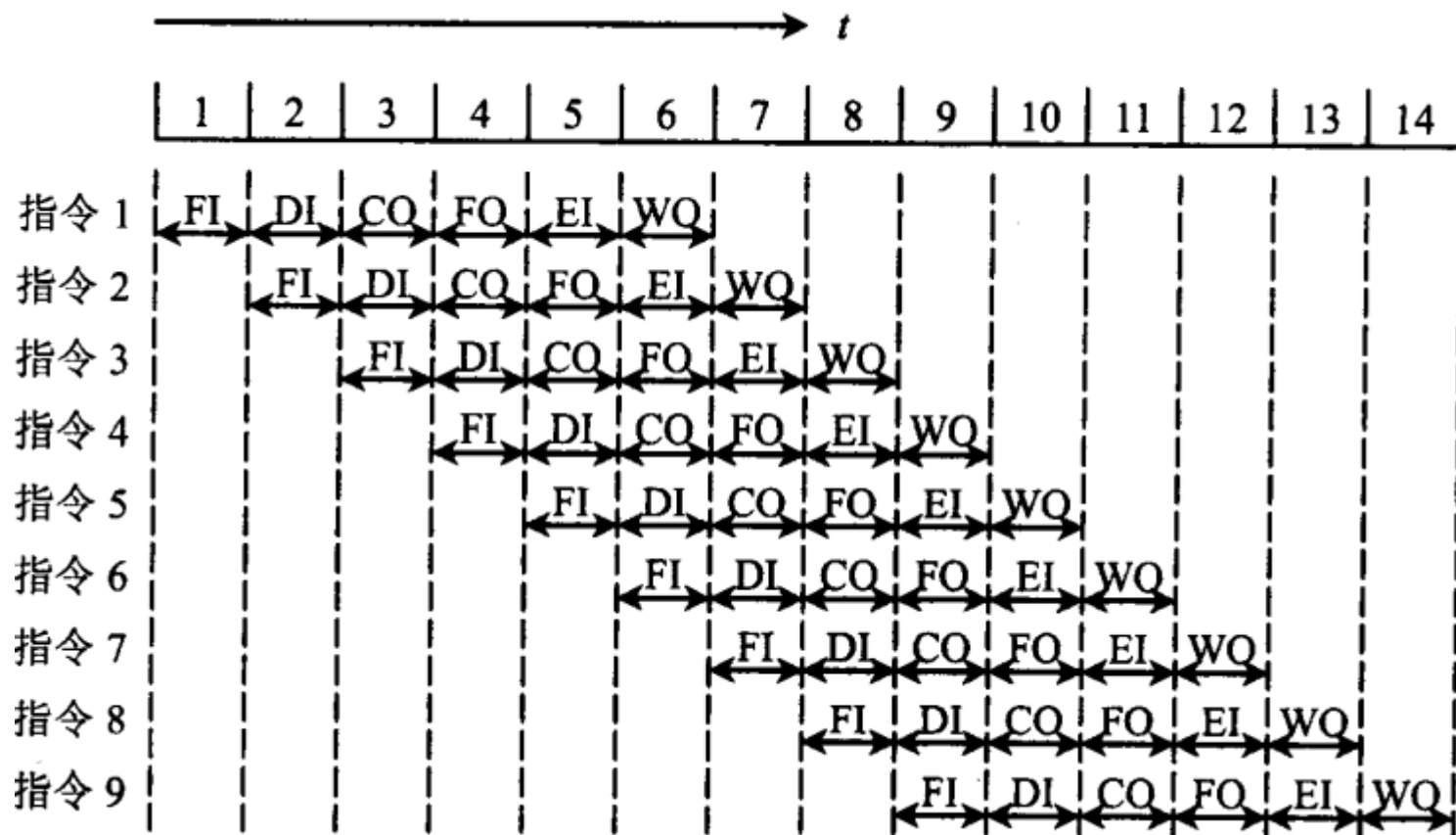
- RISC processors aim to execute one instruction per **clock cycle**.
- RISC processors break the complete fetched, decode and execution process into several **stages**.
- At any one time there will be an instruction at each stage, so for a five stage pipeline there will be five instructions **in flight**.



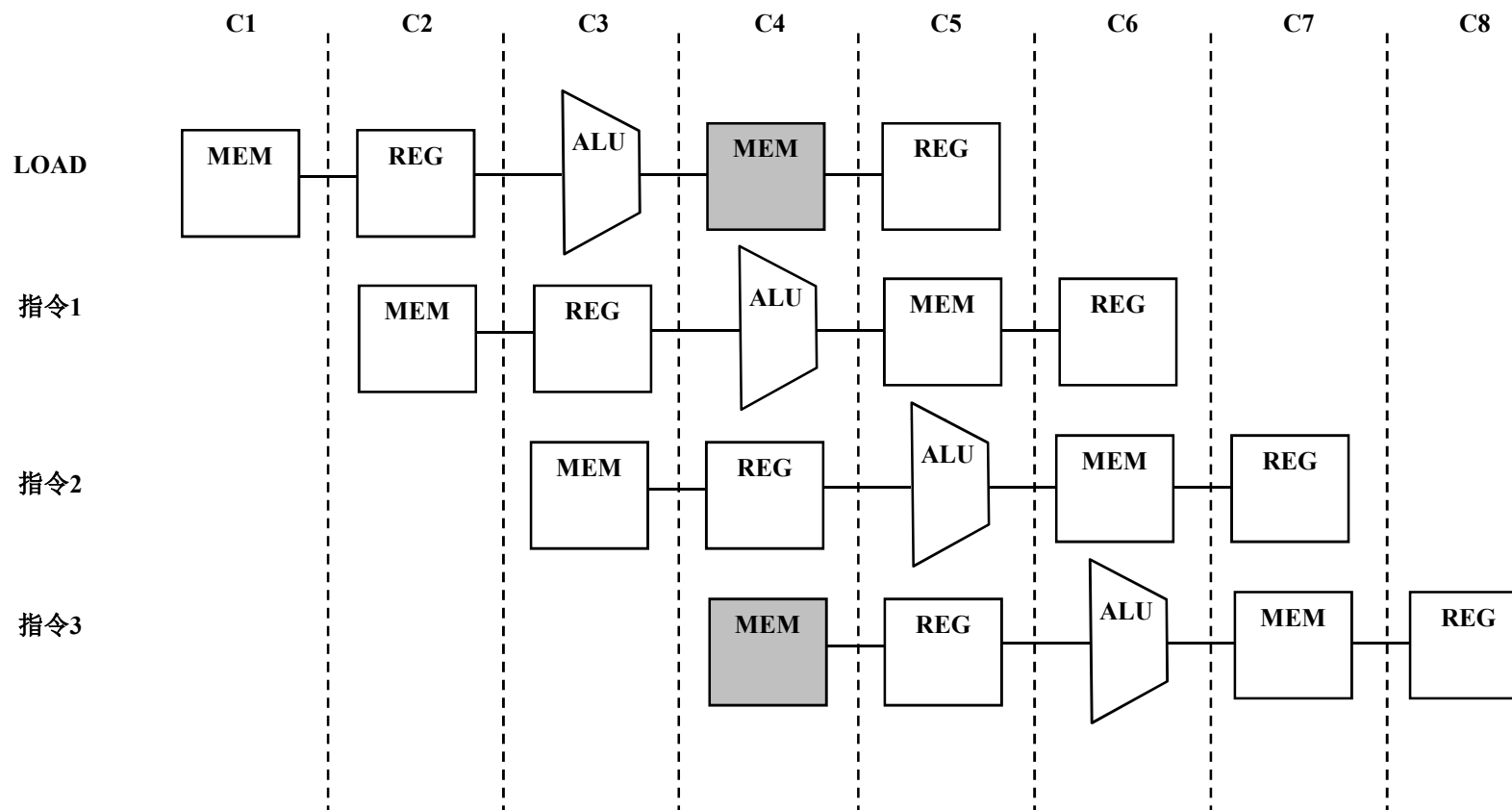
指令流水线结构与指令执行过程



指令六级流水时序(时空图)

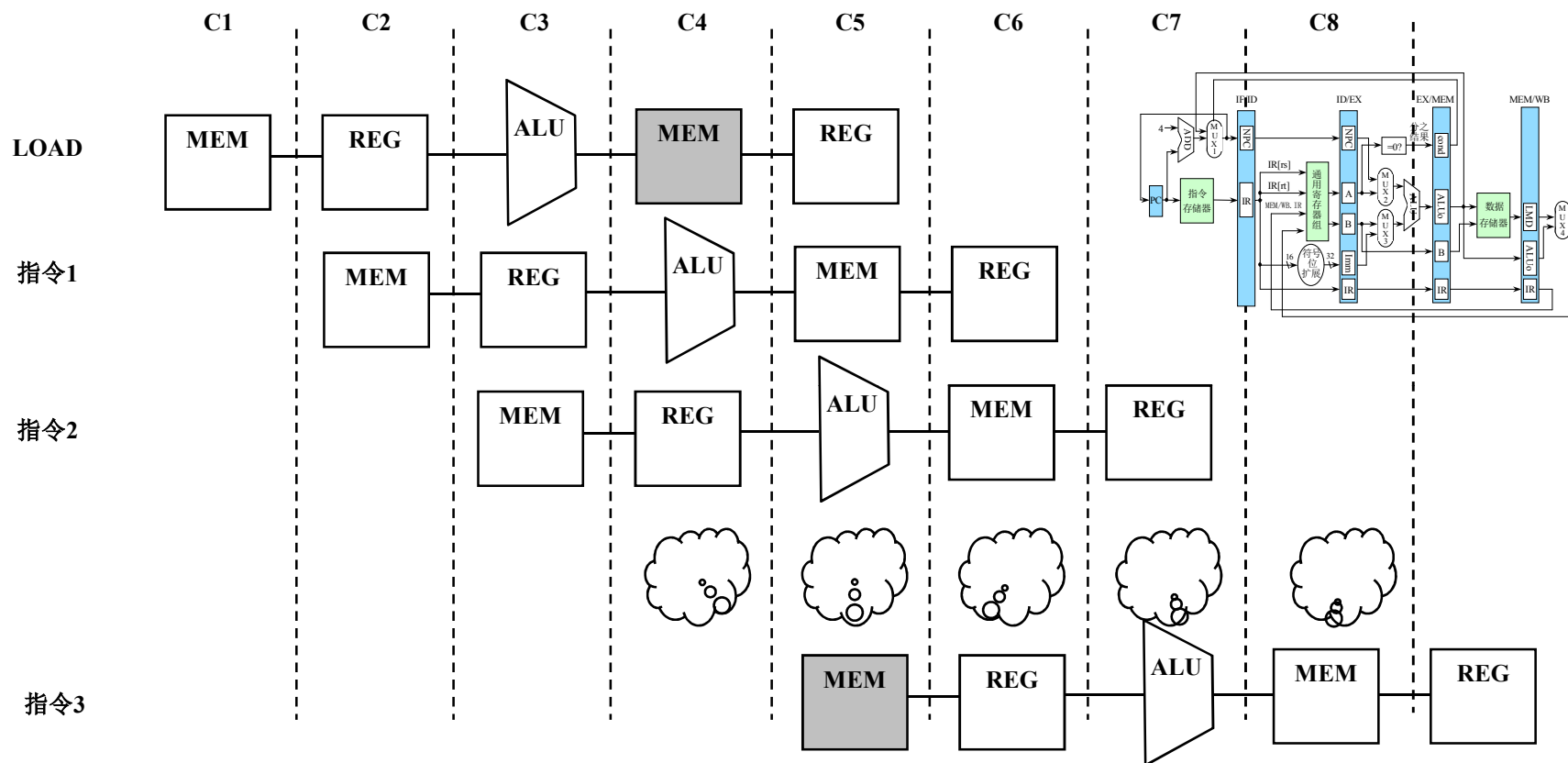


结构依赖hazard



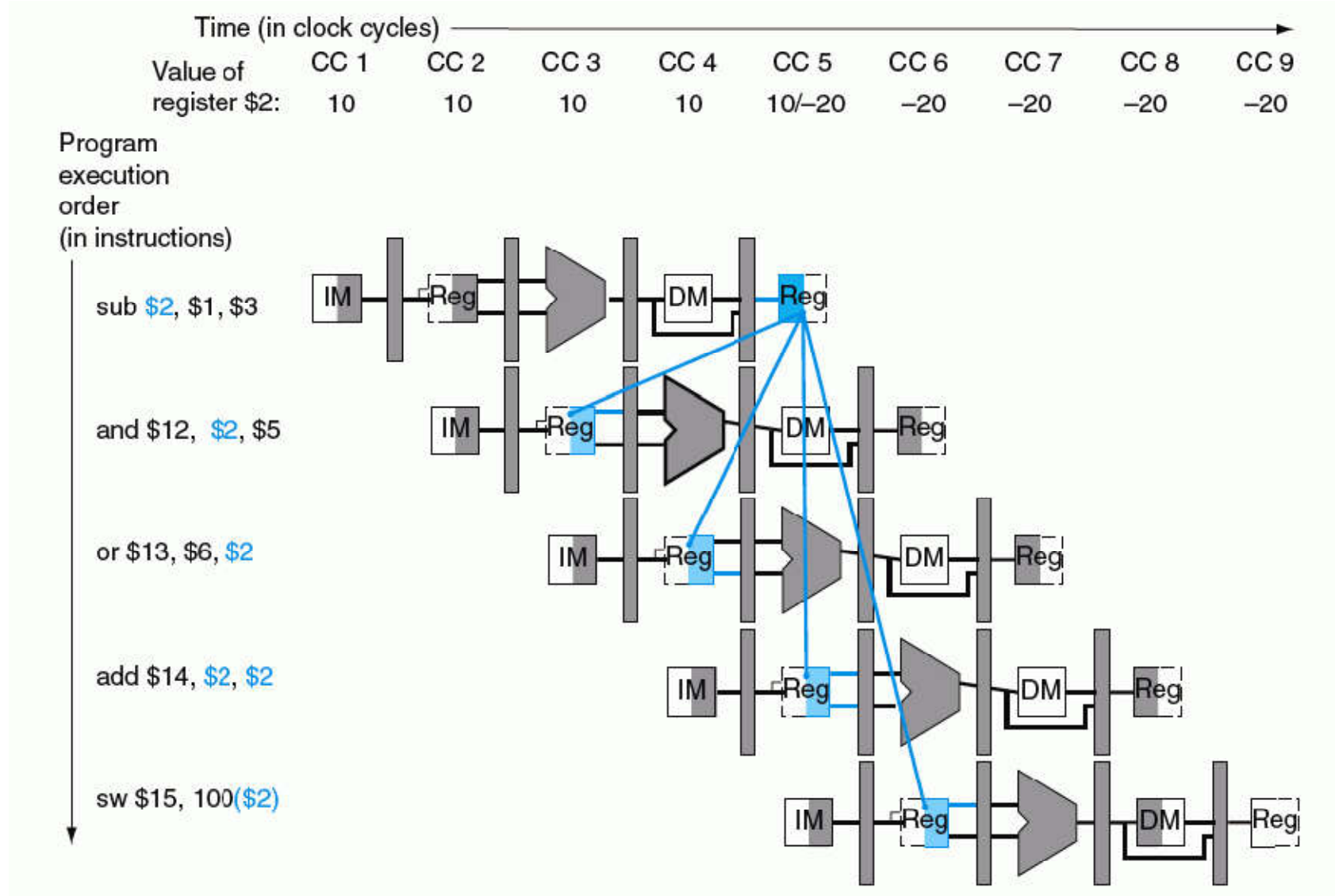
- 典型的结构相关由访存造成。

流水线停顿 (stall)



- 气泡 (bubble)：流水线停顿一个周期
 - 实现：不改变PC，重新取指，或清空流水线寄存器

data hazard(RAW)

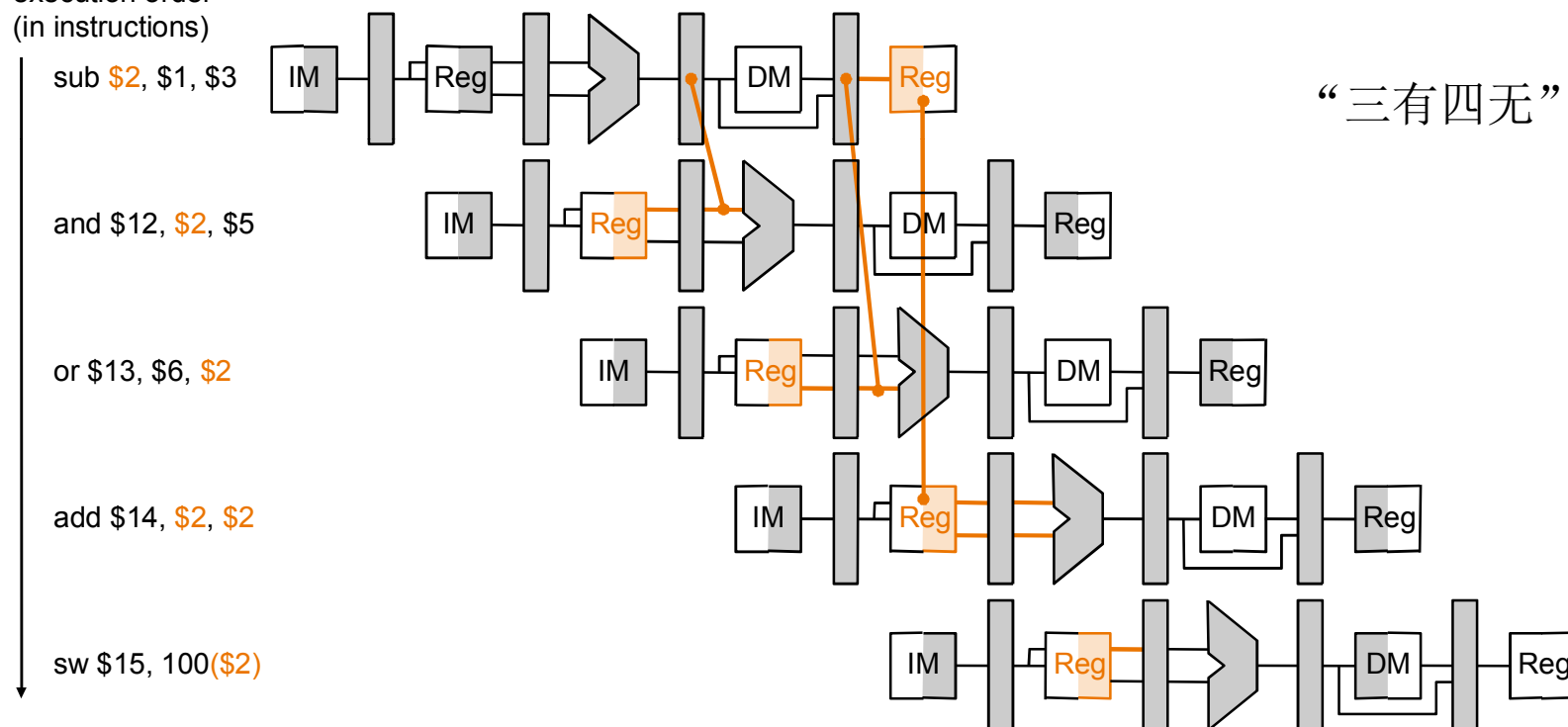




forwarding可消解的RAW

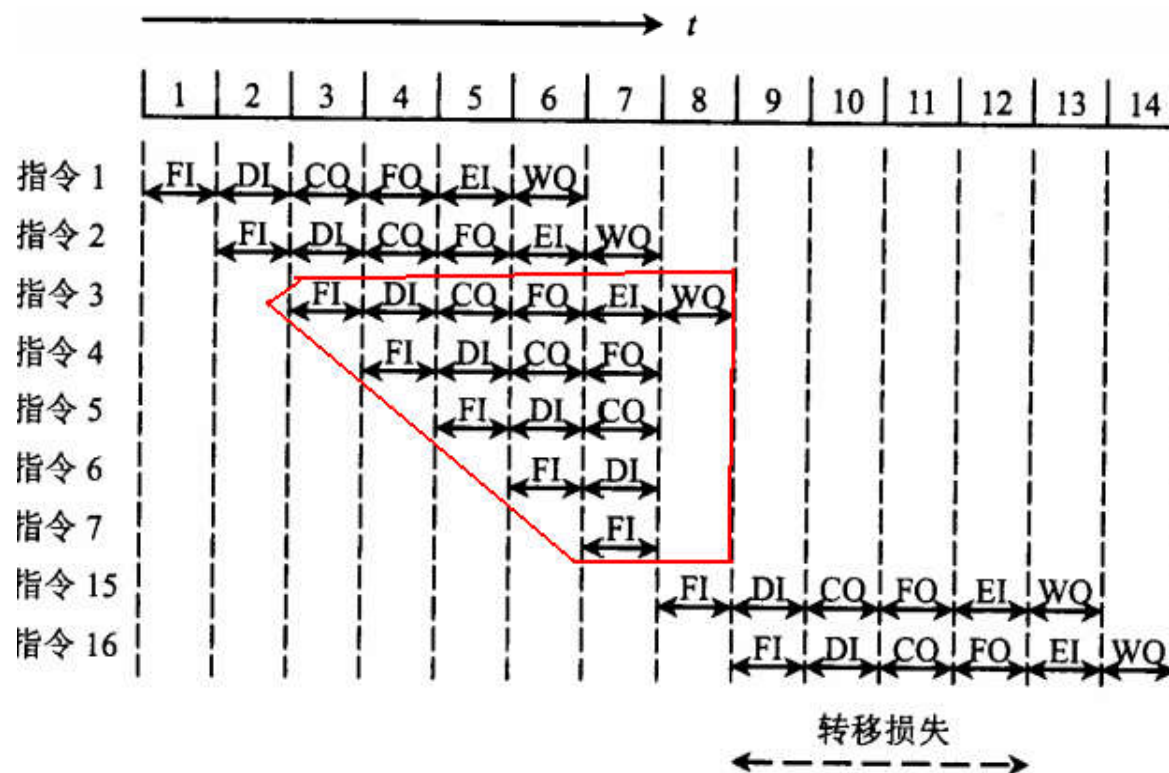
	Time (in clock cycles) →								
	CC 1	CC 2	CC 3	CC 4	CC 5	CC 6	CC 7	CC 8	CC 9
Value of register \$2 :	10	10	10	10	10/-20	-20	-20	-20	-20
Value of EX/MEM :	X	X	X	-20	X	X	X	X	X
Value of MEM/WB :	X	X	X	X	-20	X	X	X	X

Program
execution order
(in instructions)

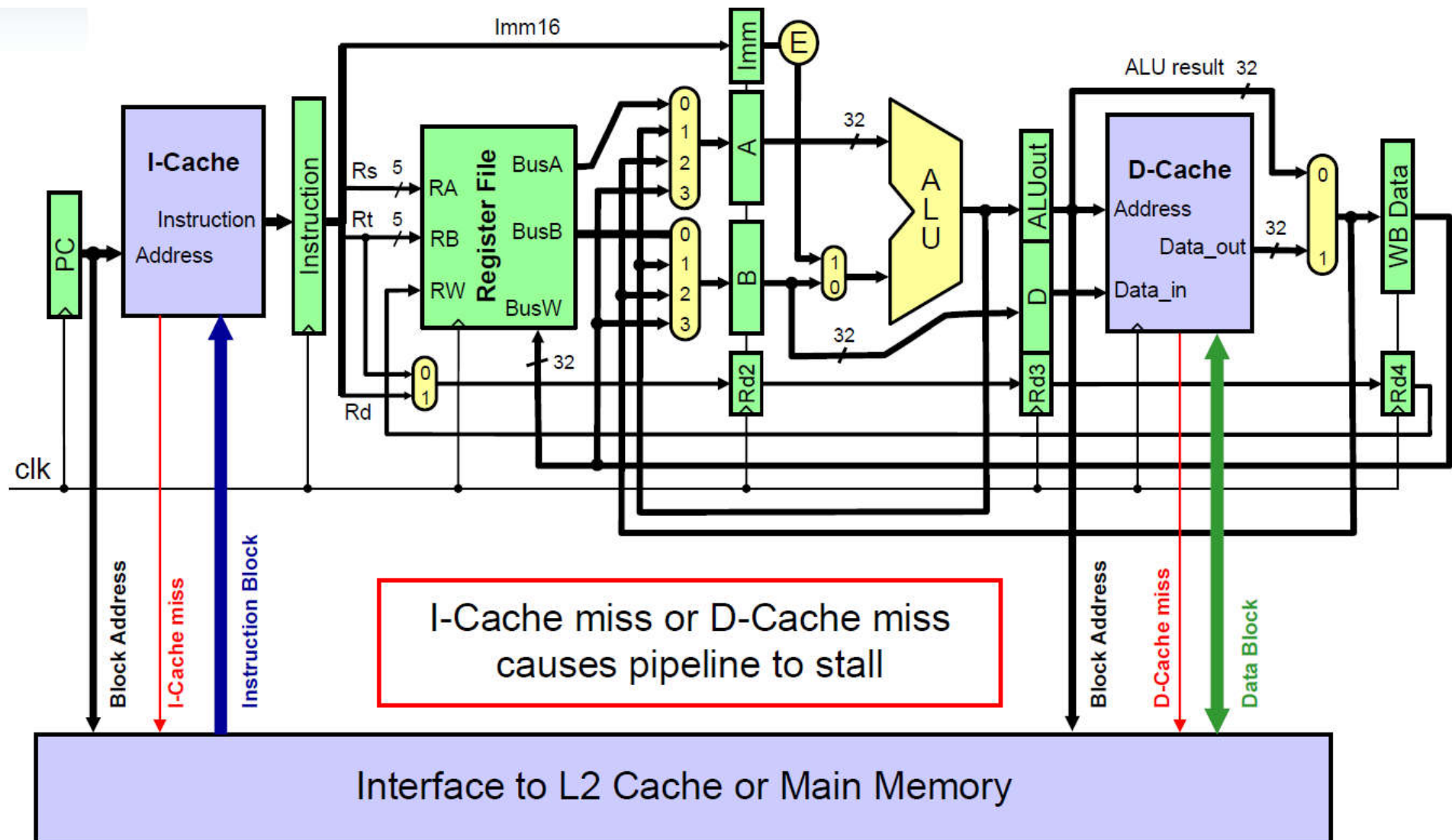


分支相关

- 假设指令3是一条条件转移指令
 - 即指令3必须待指令2的结果出现后(第7个时间单元), 才能决定下一条指令是指令4(not taken)还是指令15(taken)。
- 流水线flush
- 优化技术?

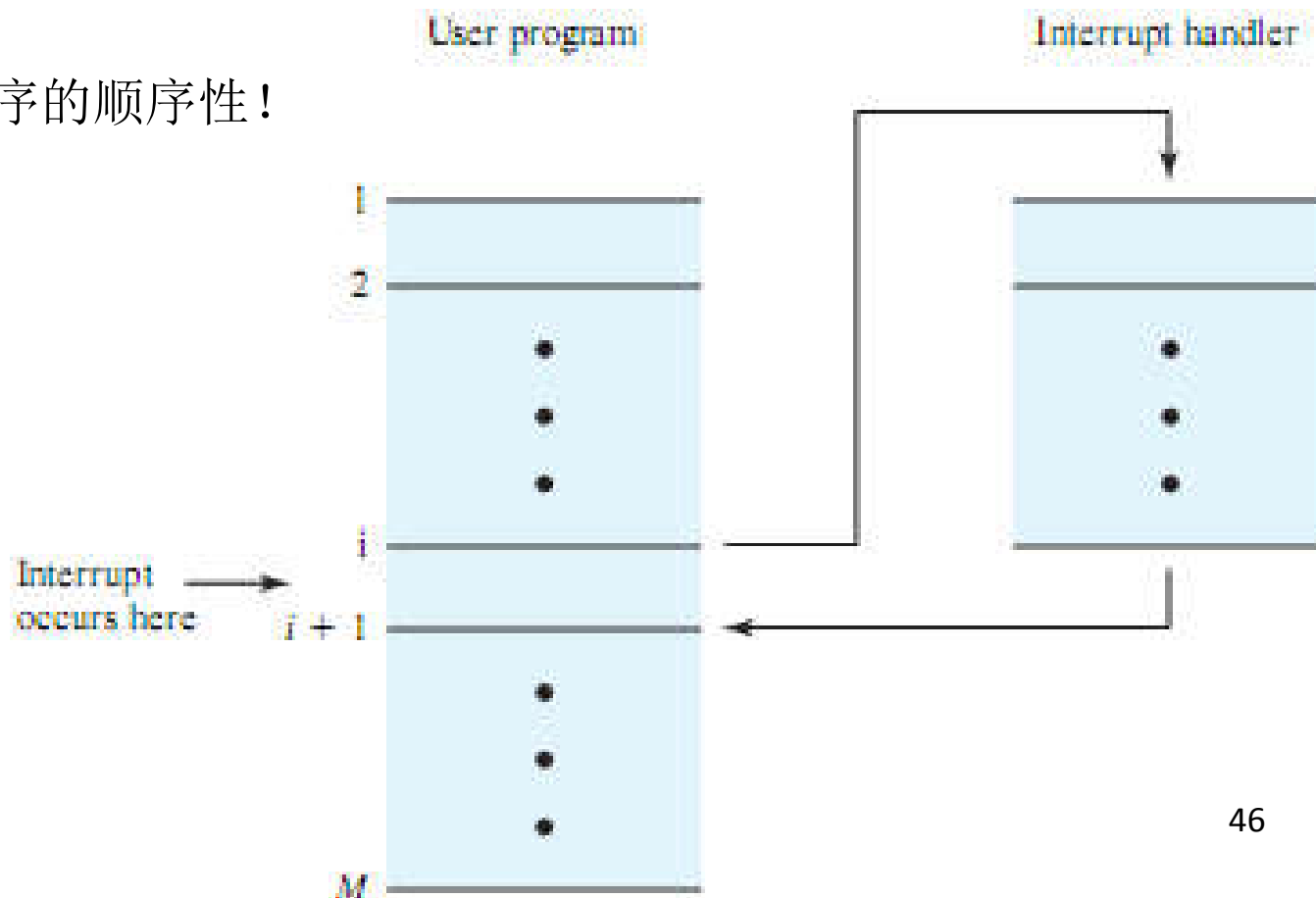
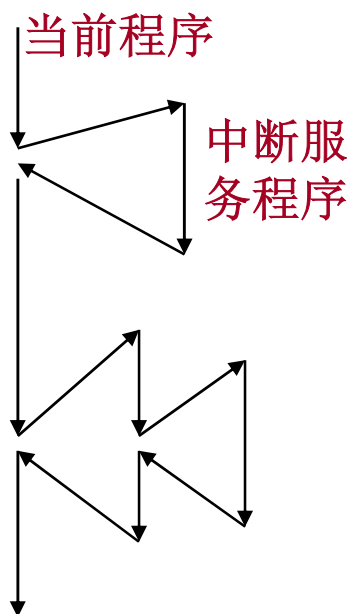


Cache miss造成流水线stall



中断与异常

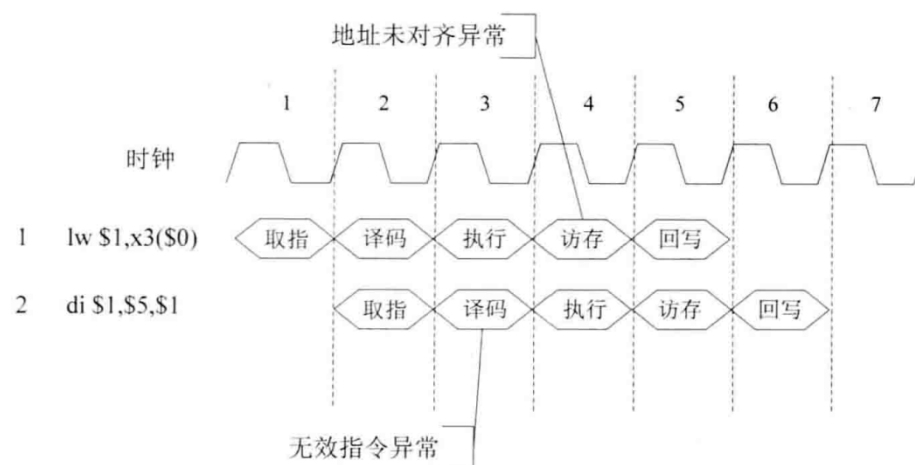
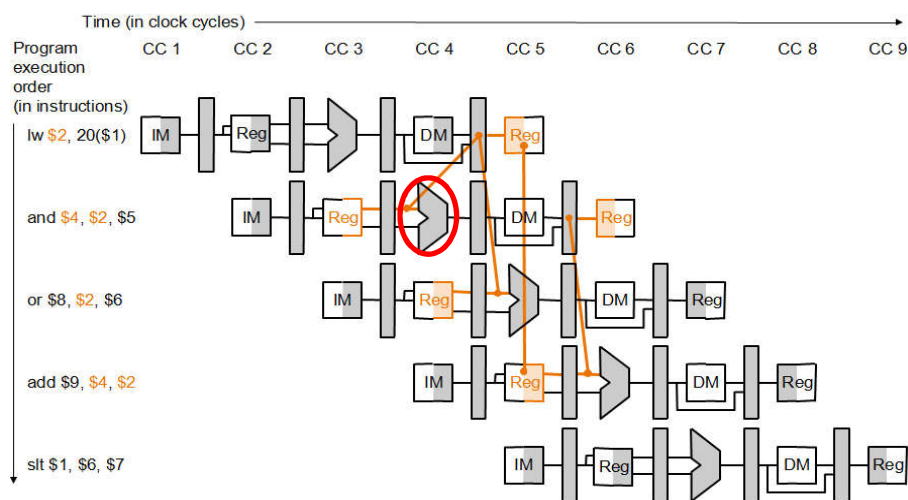
- 中断：CPU外部事件（I/O，timer，故障），异步
- 异常：CPU内部事件（访存、译码、溢出），同步
- Transfer of Control via Interrupts的过程
- 中断嵌套
- 断点：保证程序的顺序性！





流水线的异常处理

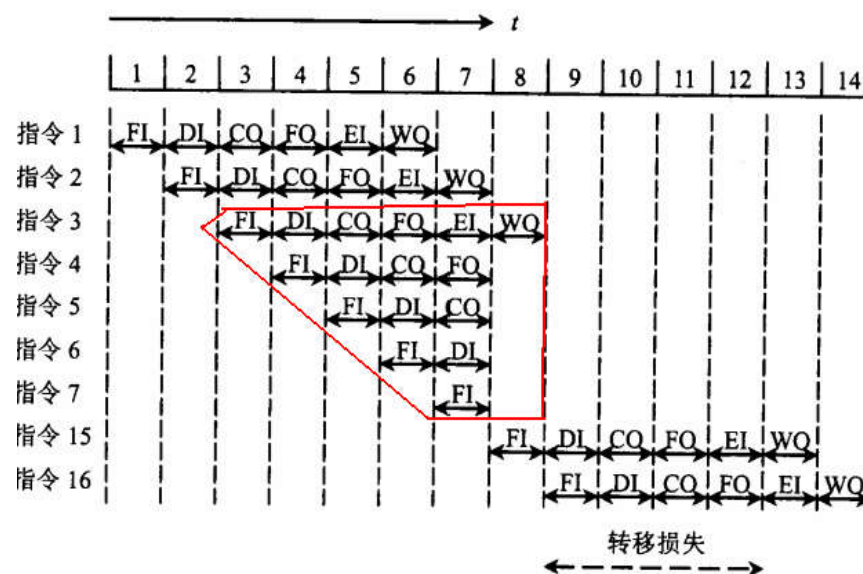
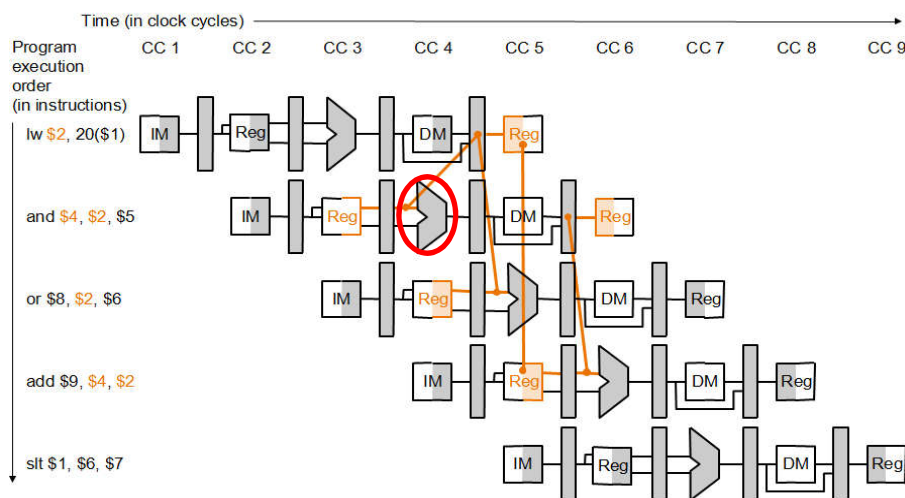
- 保证程序的顺序性，需要“精确”定位“当前指令”
 - 左图：**and**溢出
 - “精确”语义为：前面指令完成，后续指令不执行！
- 问题
 - 并发：多个流水段，多条指令，多种异常（左图，CC4）
 - 异常发生顺序与指令执行顺序不一定相同（右图）
 - 后一条指令先发生异常，前一条指令后发生





流水线的异常处理（续）

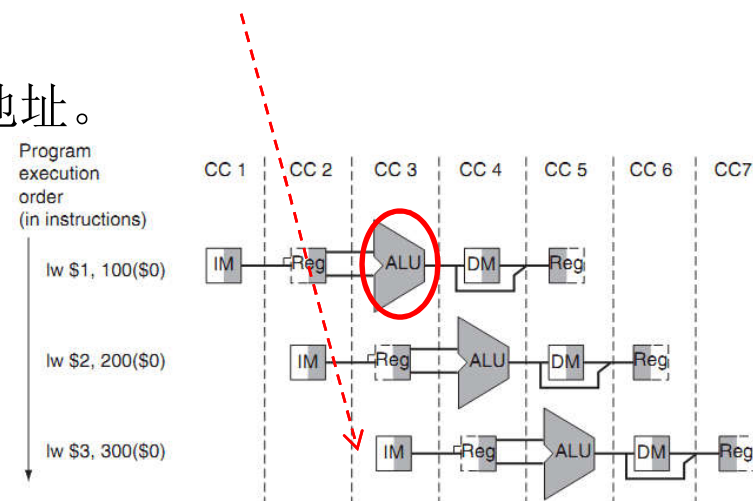
- 保证程序的顺序性，需要“精确”定位“当前指令”
 - 如果and出现溢出，“精确”要求前一条指令执行完，后续指令不执行！
- 问题
 - 后续指令在产生异常指令完成之前改变了某些系统状态（左图）
 - ld 指令在MEM段产生异常，而其后的R-type指令设置了0标志
 - 转移预测失误可能取消其后出现了异常的指令



非精确异常策略



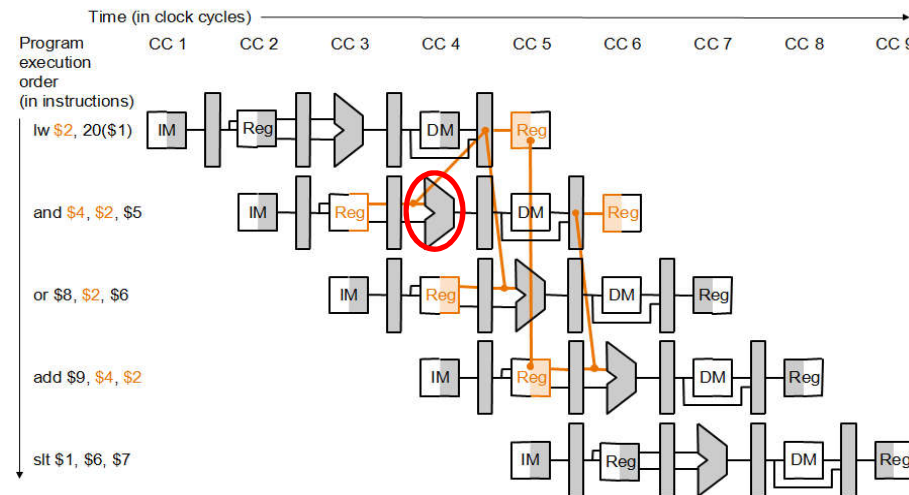
- 允许**已进入**流水线中的指令执行完再转去执行中断处理
 - 无论在第*i*条指令的哪一流水段上发生异常, 都不再允许后继指令进入流水线
 - 断点: **最后**进入流水线的那条指令的地址。
- 优点: 硬件比较简单
- 缺点:
 - 异常响应时间**抖动**
 - 可能会导致程序**出错**:
 - *i*: `FADD R1, R2; (R1)+(R2) -> R1`, 如果此时溢出?
 - *i*+1: `FMUL R3, R1; (R3)×(R1) -> R3`, 无效执行!
 - *i*+1以进入流水线, 需要执行完, 但结果无效!
 - 注意, 与RAW无关
 - 程序调试不便:
 - 程序员在第*i*条指令设置断点, 但程序不能准确中断在所设置的断点处。



精确异常语义



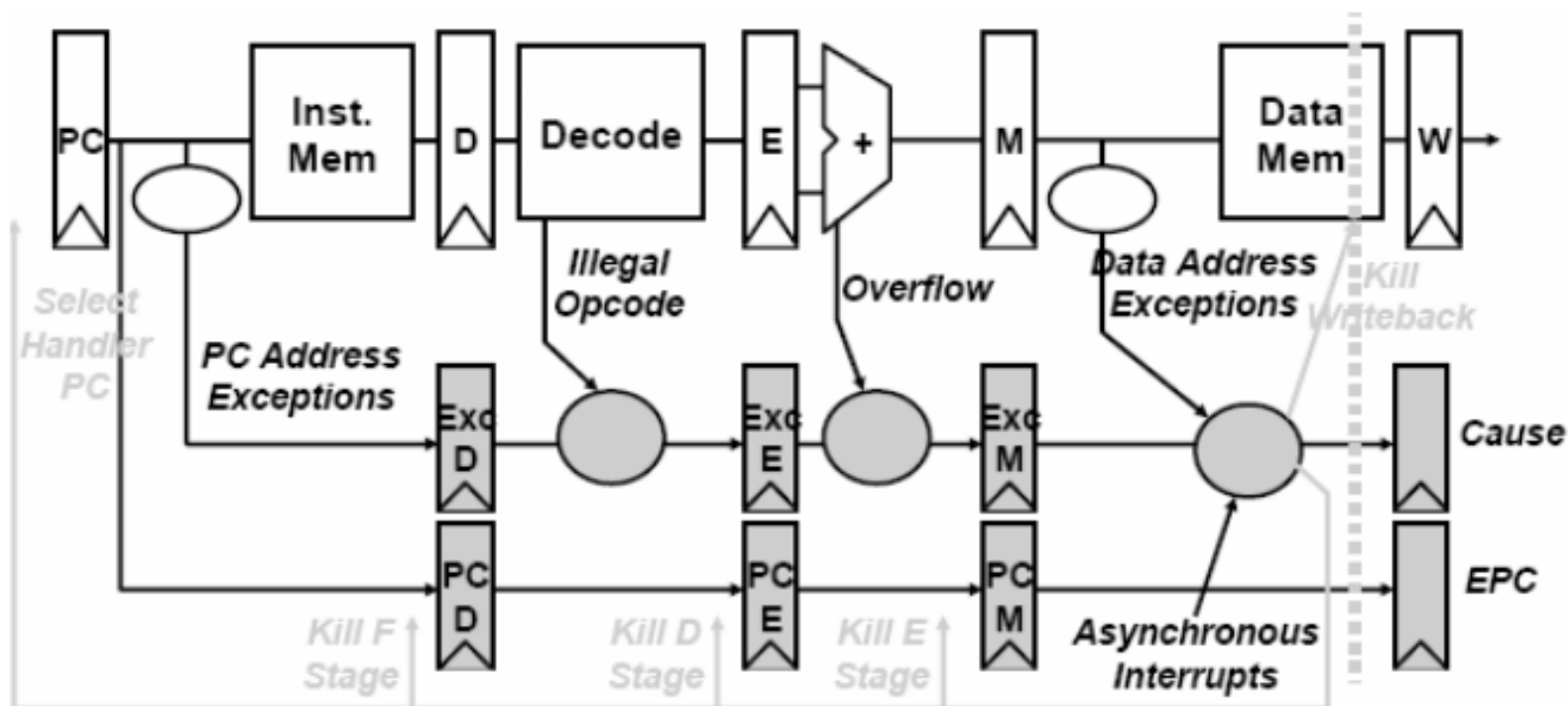
- **Case 1:** 发生在两条指令之间(中断)
 - 前一条及其之前的指令都已执行完成，后一条指令还没有启动。
- **Case 2:** 发生于单条指令(异常)，则当且仅当满足：
 - 所有之前的指令都已经提交其状态
 - 所有后续指令（包括中断的指令）没有改变任何机器的状态



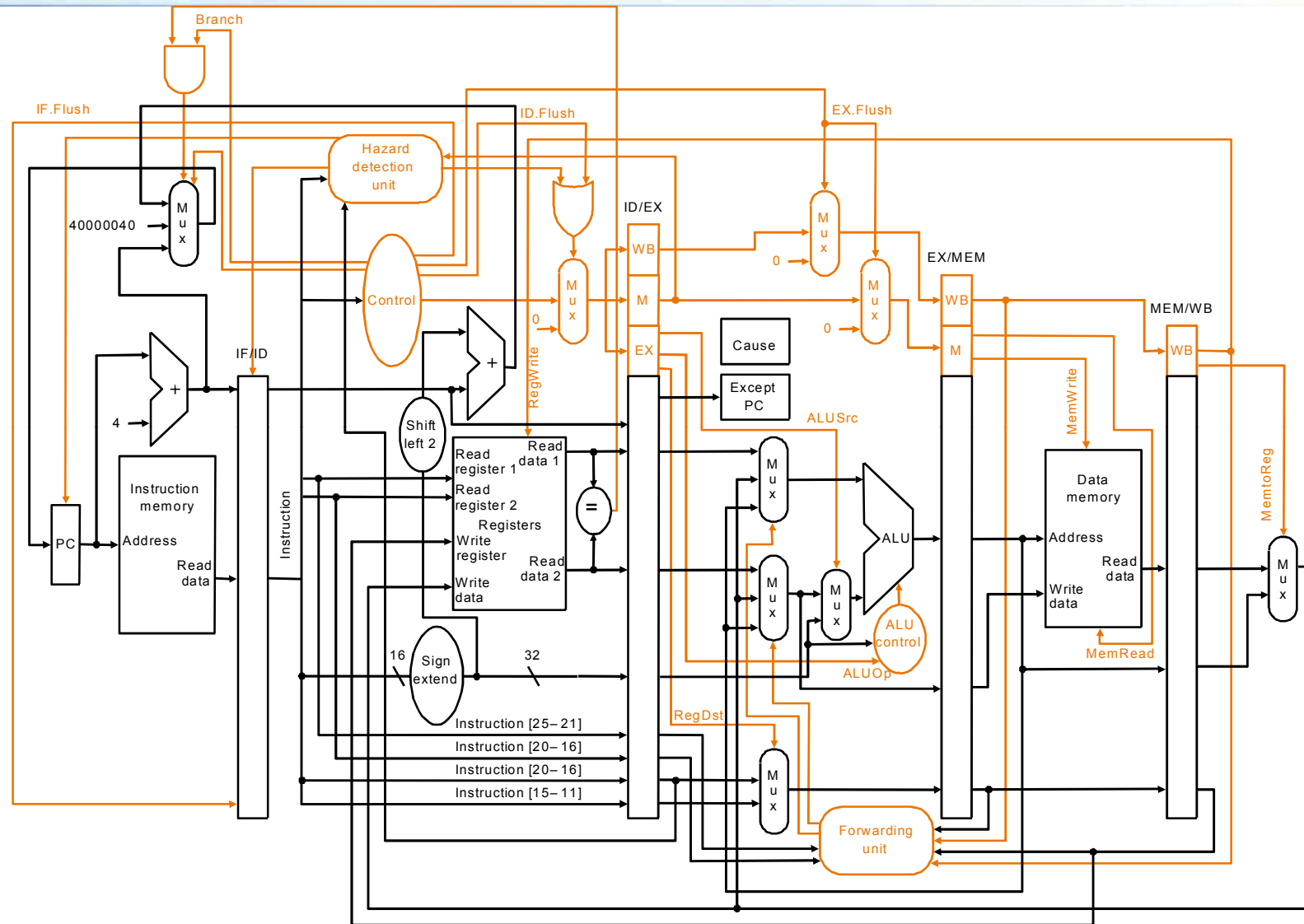
各段产生的异常及处理：MIPS的策略



- 保持流水线的异常标记直到**提交点**（M段）
- **早**期流水段的异常抑制**后**来的异常（flush IF/ID/EX）
- 提交点处引入**外部中断**（**抑制**其他异常）
- 如果提交点有异常，则更新**cause**和**EPC**，清除所有流水段，回复PC值到**fetch**段

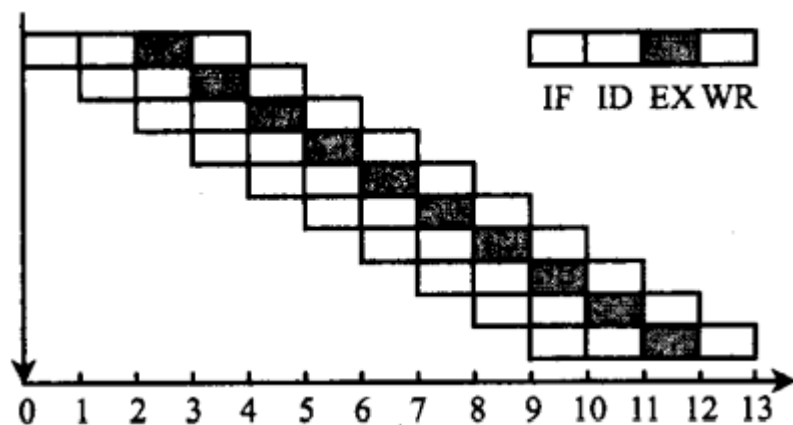


MIPS处理器的五级流水结构

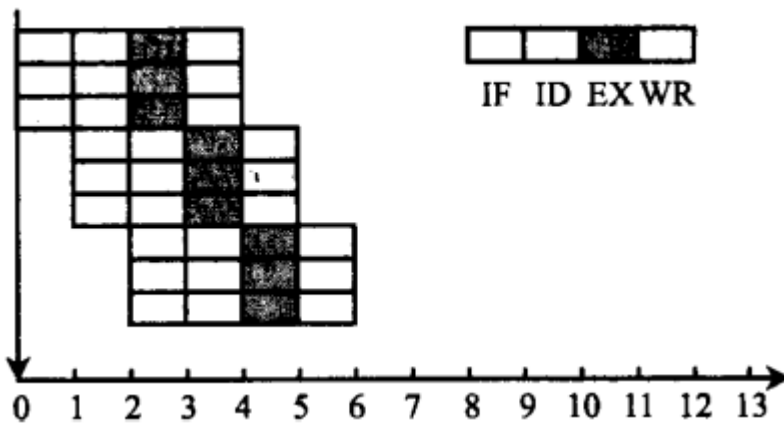


op(6 bits)	rs(5 bits)	rt(5 bits)	rd(5 bits)	shamt(5 bits)	funct(6 bits)
------------	------------	------------	------------	---------------	---------------

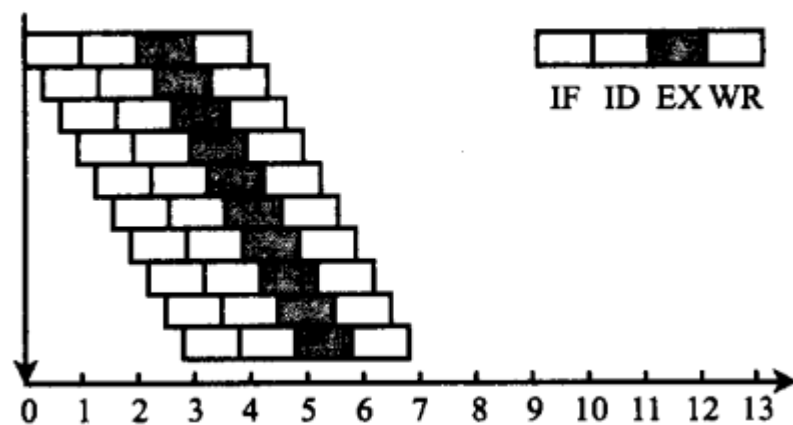
四种ILP技术比较



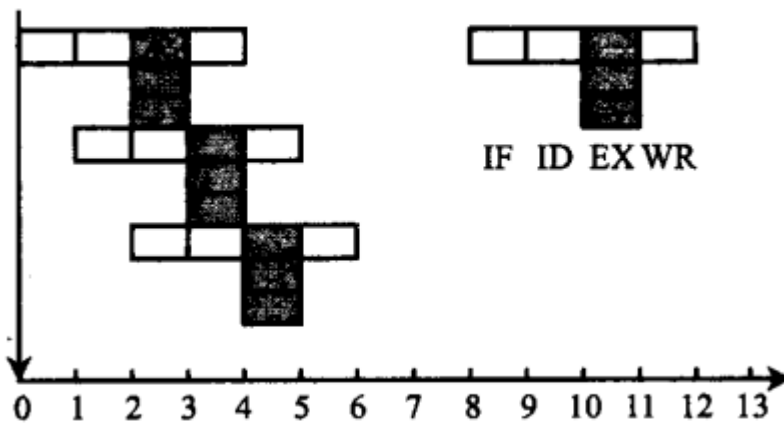
(a) 普通流水



(b) 超标量流水

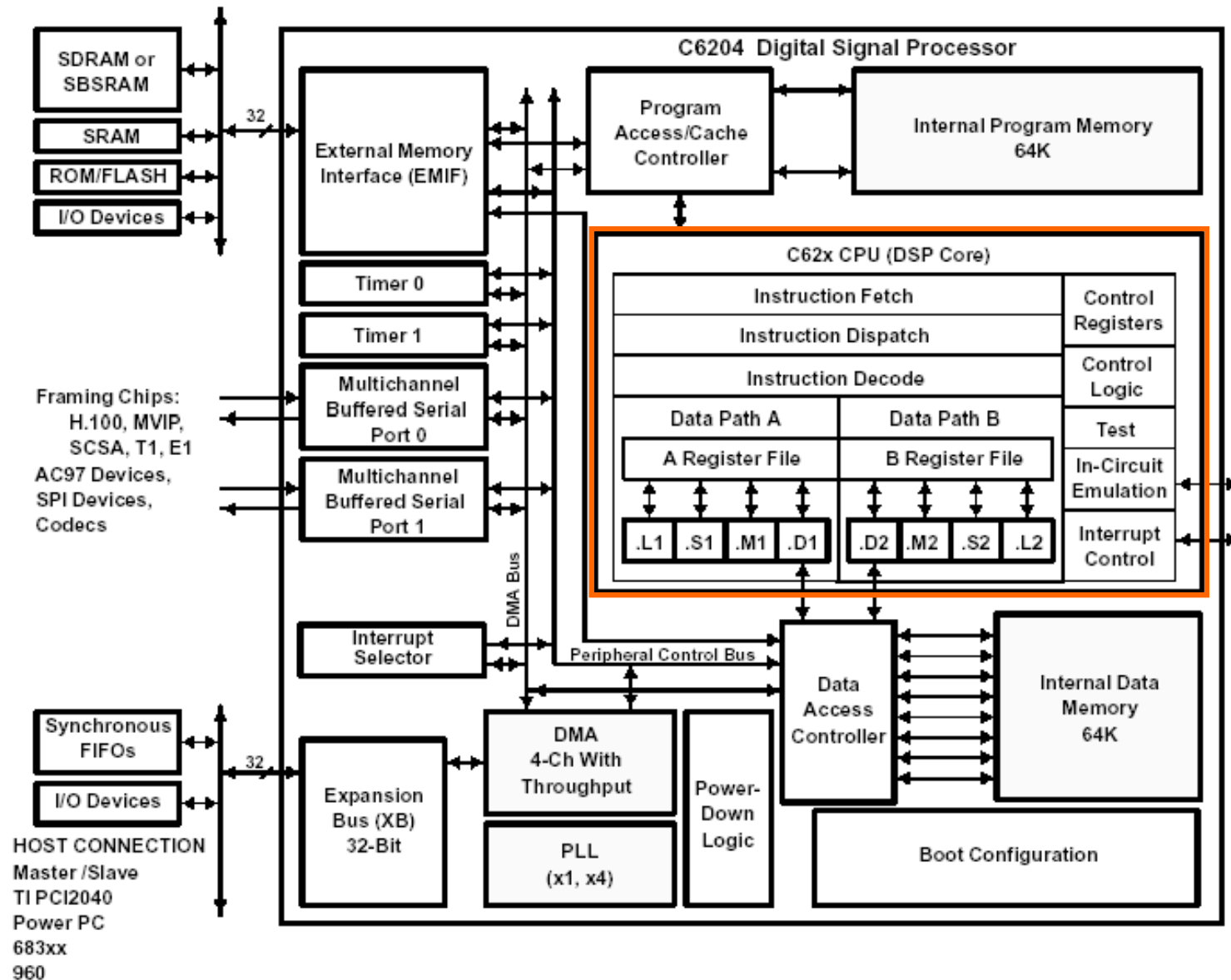


(c) 超流水线

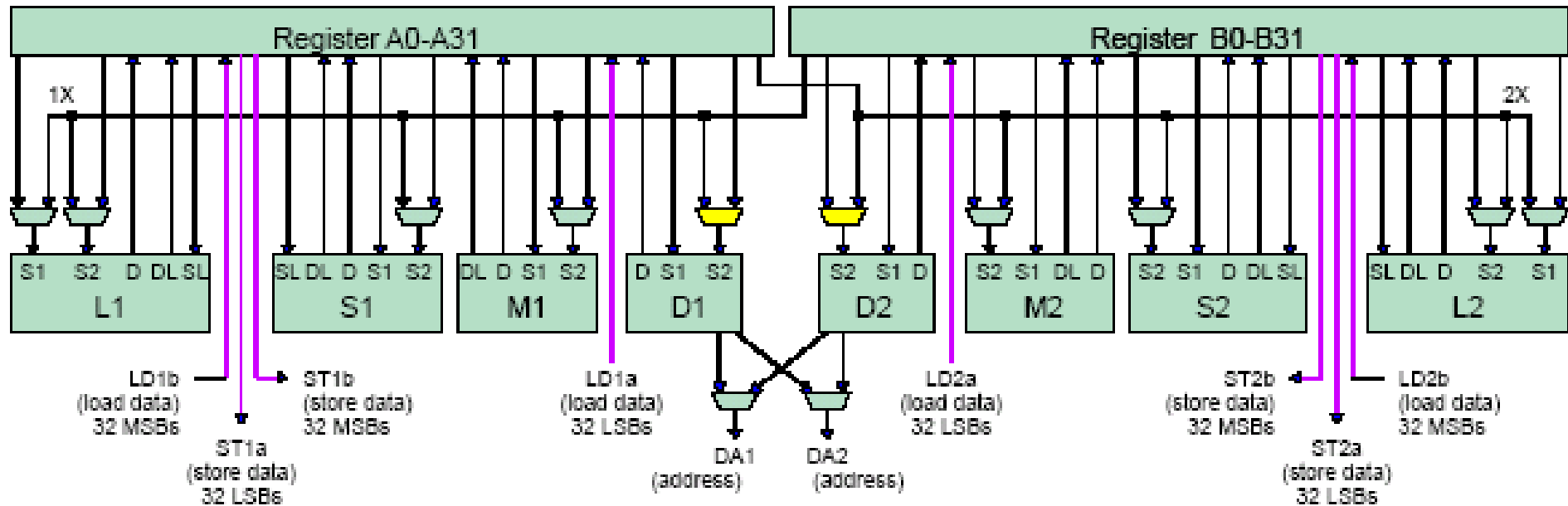


(d) 超长指令字

TMS320C64x: VLIW



TMS320C64x: 数据通路

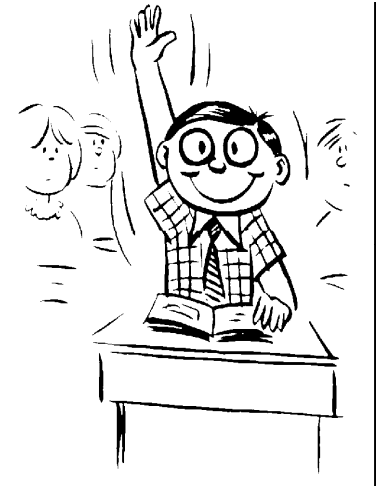


- .M单元主要完成乘法运算，.D单元是唯一能产生地址的功能单元
- .L与.S是主要的算术逻辑功能单元（ALU）
- Two general-purpose register files (A and B) $64 \times 32\text{bit}$
- Two load-from-memory paths (LD1 and LD2) 64bit
- Two store-to-memory paths (ST1 and ST2) 64bit
- Two register file cross paths (1X and 2X)
- Two data address paths (DA1 and DA2)

ISA: A Minimalist Perspective



- **ISA design decisions must take into account:**
 - **technology**
 - **machine organization**
 - **programming languages**
 - **compiler technology**
 - **operating systems**
- 最小的计算机系统由哪些部件构成？
- 最小的ISA需要哪几条指令？
- 需要哪些寻址方式？



OISC: the one instruction set computer



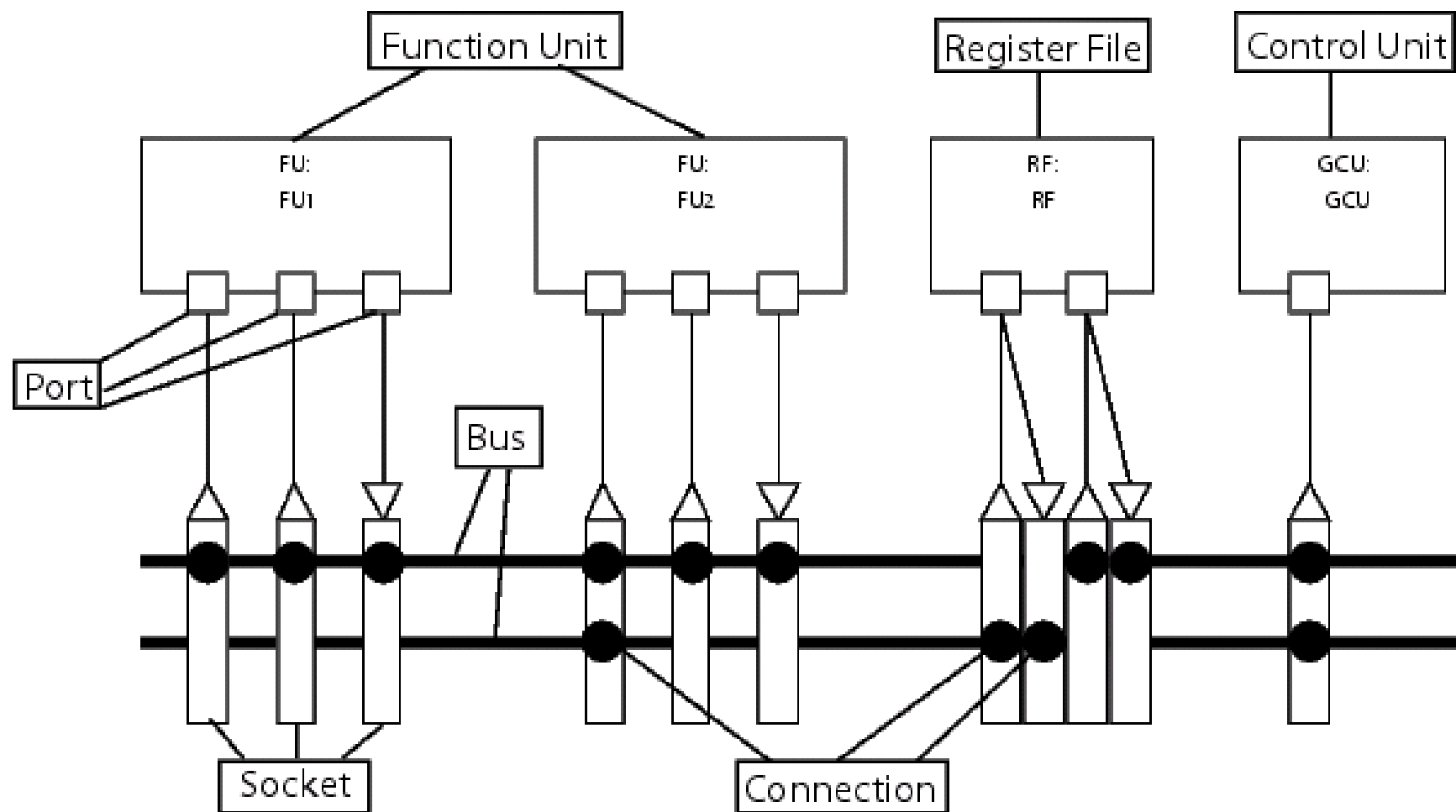
- 以下两条之一
 - SBN指令: subtract and branch if negative
 - URISC体系: the ultimate reduced instruction set computer
 - Move指令:
 - TTA体系
- OISC的优势
 - 硬件极其简单
 - 程序员有充分的控制权
 - 优化由编译器完成
 - 灵活
 - 其他“指令”都可由该指令构造
 - 意味着用户可自定义指令集
 - 意味着可适用于任何领域（构造ASIP）
 - 低功耗
- 应用：嵌入式处理器

Transport triggered architecture



- uses only the **MOVE** instruction
 - originally called a "move machine"
 - 程序直接控制处理器内部的传输总线
 - 计算：由memory-mapped ALU完成
 - 计算操作伴随数据传输完成
 - 向功能单元的触发端口写数据将触发功能单元完成计算
 - 跳转：由memory-mapped PC完成
 - 适于实现ASIP
- 上市产品：MAXQ微控制器
 - 适合数模混合应用场景

TTA的典型结构

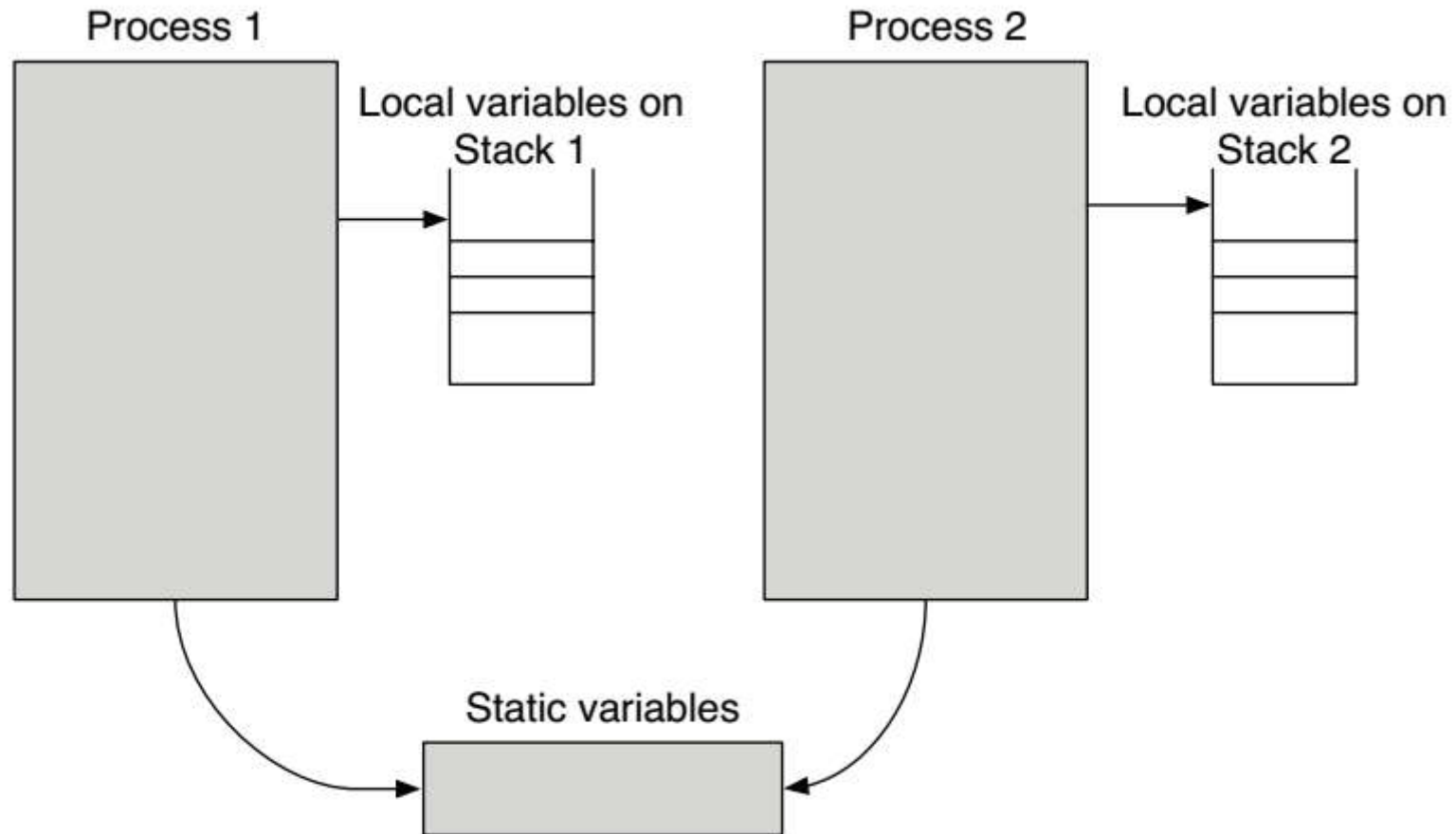




存储器

- SRAM: 一bit大约需要六个晶体管, 读写速度较快。
- DRAM: 一bit大约需要一个晶体管和一个电容, 需要刷新。
- SDRAM: 与CPU频率同步, 共享一个时钟周期。
- FLASH存储器: 既具有SRAM读写的较快的访问速度, 又具有ROM在断电后信息不丢失的特点。由于FLASH不需要存储电容器, 故其集成度更高, 制造成本低于DRAM。
 - NOR技术Flash
 - 拥有独立的数据总线和地址总线, 能快速随机读取
 - 重新编程之前需要对块或整片进行预编程和擦除操作
 - 擦除和编程时间长, 不适于纯数据存储和文件存储的应用
 - NAND技术Flash
 - 具有快编程和快擦除的功能, 其块擦除时间比NOR技术快一百多倍
 - 数据、地址采用同一总线。随机读取速度慢且不能按字节随机编程。
 - 芯片尺寸小, 成本(bit cost)低, 可取代硬盘或其他大容量存储设备

C程序空间

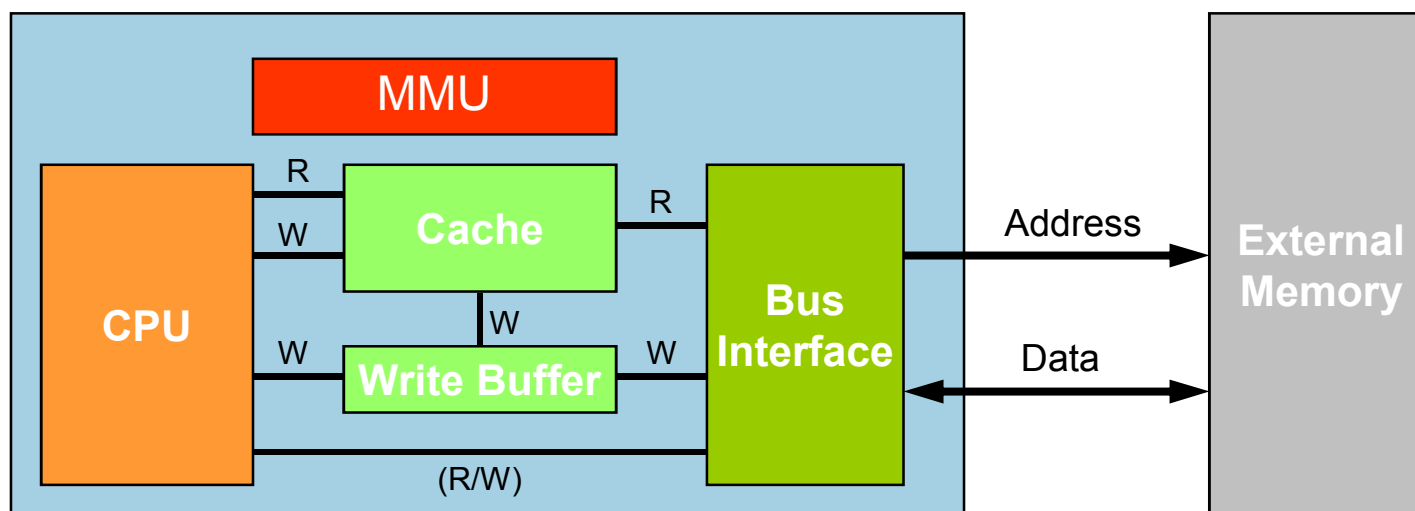


- Ctr-C, Ctr-V?

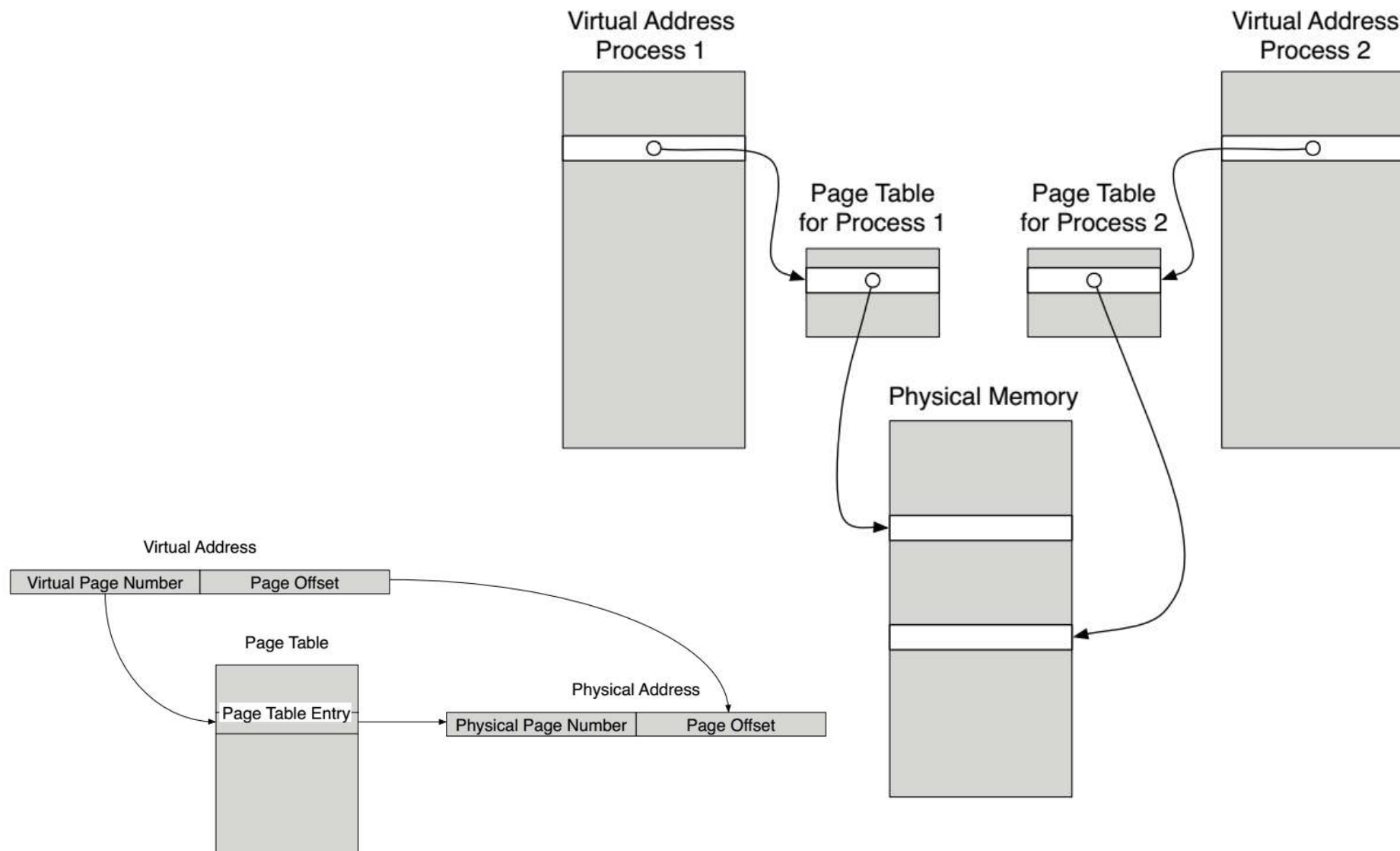


高效的存储器系统

- Cache和Write Buffer提高系统存储器访问性能
- MMU支持虚拟/物理地址映射，满足操作系统内存管理需要
- MMU中的地址转换页表和Cache的内容需要按需更新
 - 带有一定的随机性，造成某些实时不确定性
 - ARM的MMU和Cache设计允许部分随机内存锁定，以提高实时性能



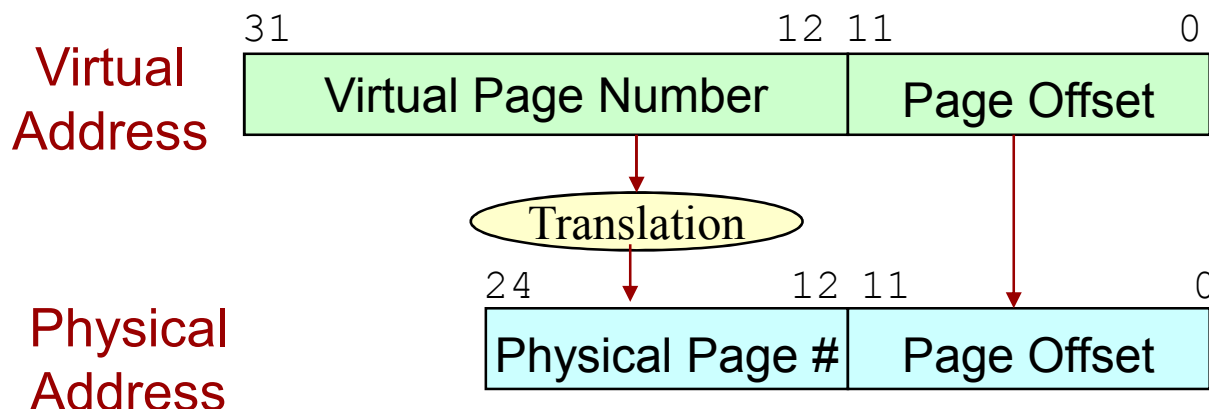
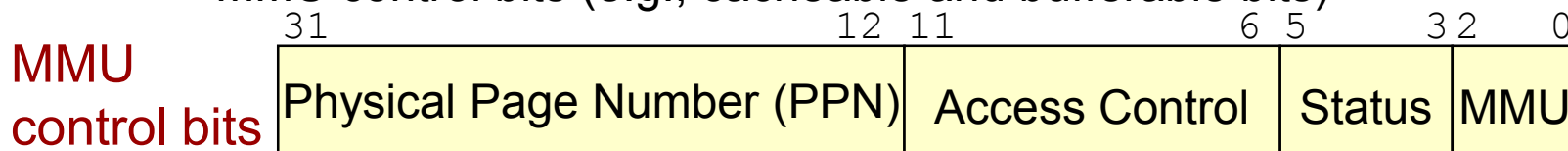
多任务存储空间管理





内存虚实映射

- 一个虚拟地址由虚拟页号和页内偏移量两部份组成。
 - 如果页的大小是 4KB，则虚拟地址的0至11位是偏移量，第12位以上是虚拟页号。
- MMU使用页表将虚拟页编号转换到物理的页，并根据页内偏移量访问物理页的正确偏移处。
 - 每一个页表项(PTE)包括以下信息：
 - 本页表条目描述的物理页编号
 - 访问控制位(是否可以写?)
 - 有效标志：表示页表本条目是否有效
 - MMU control bits (e.g., cacheable and bufferable bits)

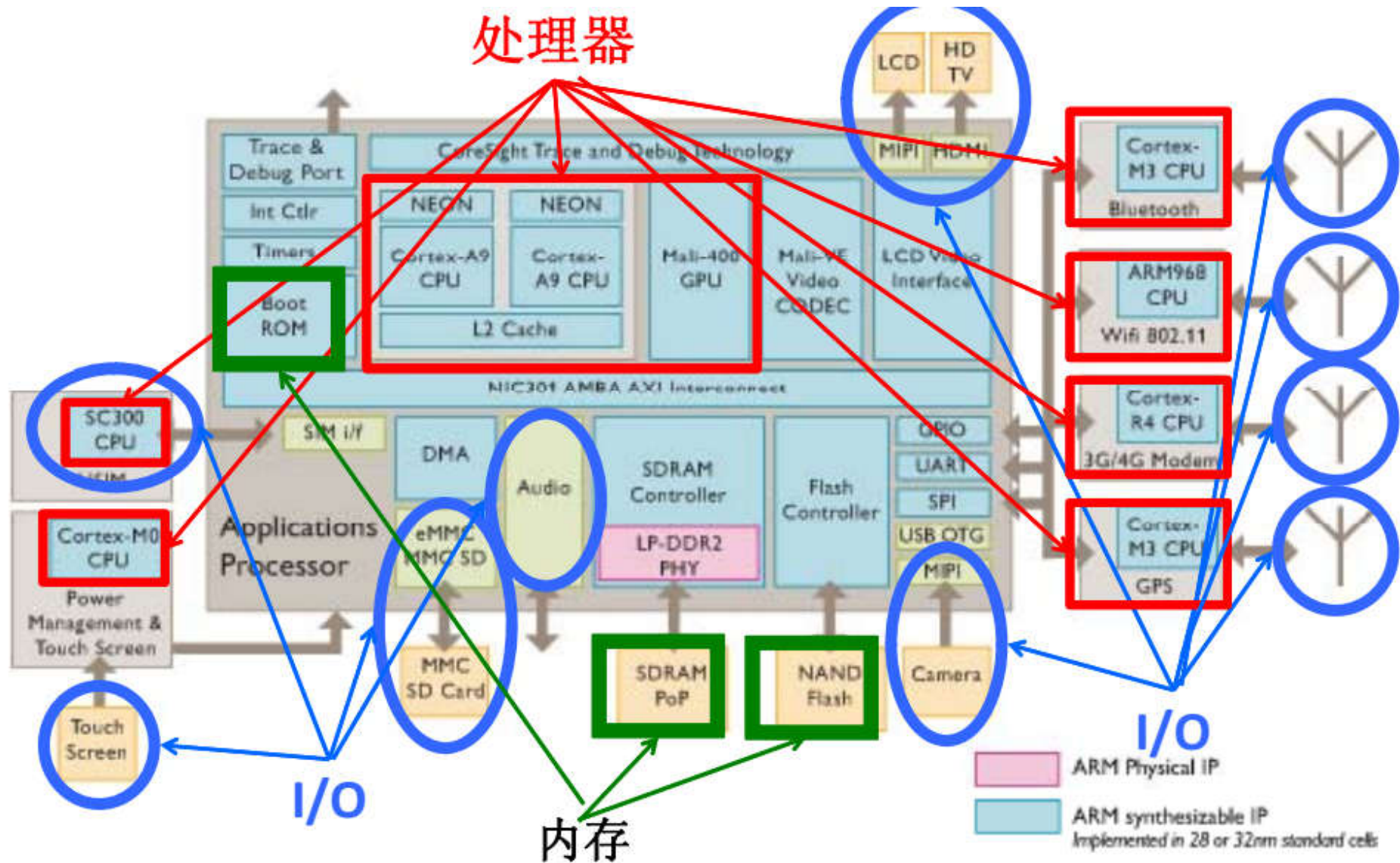


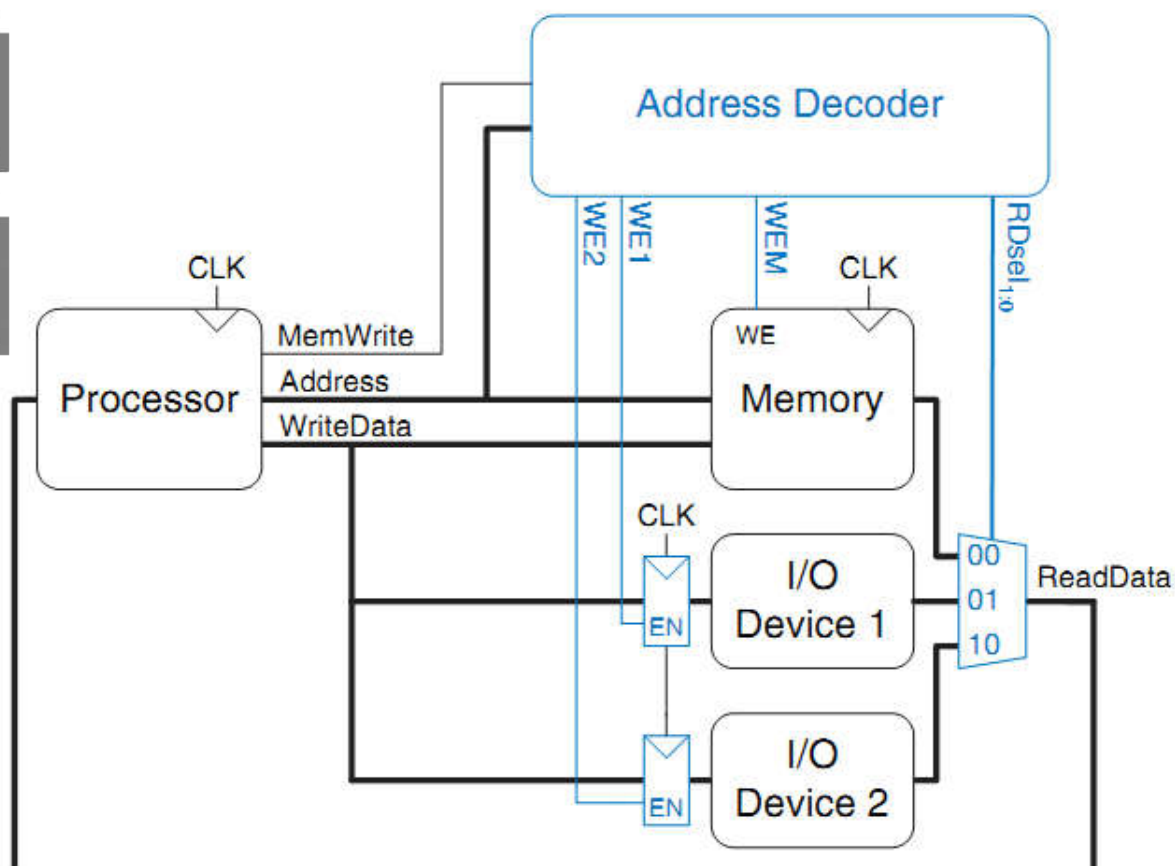
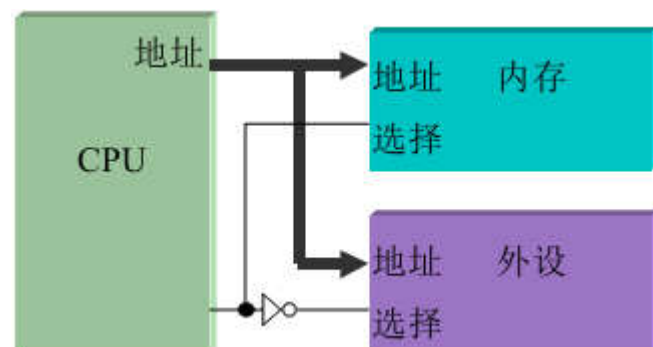


提升MMU性能

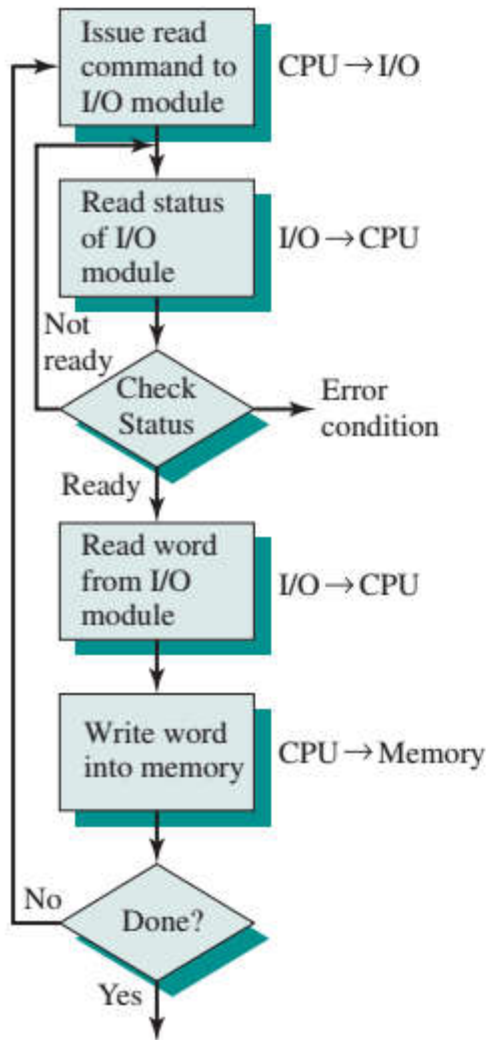
- **TLB, Translation Lookaside Buffers**
 - 存放部分页表 (**PageTab, translation table**)
 - 含: 虚址及其对应的物理地址, 权限, 域和映射类型。。。
- 当**CPU**对一虚拟地址进行存取时
 - 首先搜索**TLB**表以查找对应的物理地址等信息
 - 如果没有查到, 则查找**translation table** (称为 **Translation Table Walk, TTW**)。
 - 经过**TTW**过程后, 将查到的信息保存到**TLB**。
然后根据**TLB**表项的物理地址进行读写。

iPhone

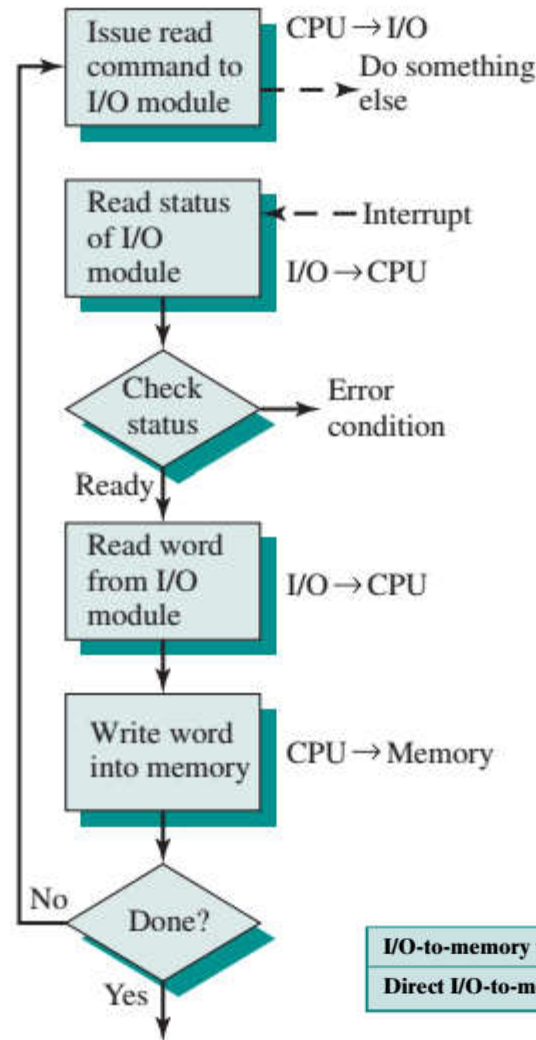




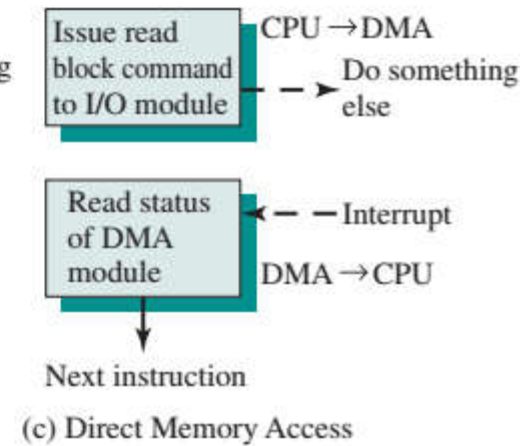
Three Techniques for Input of a Block of Data



(a) Programmed I/O

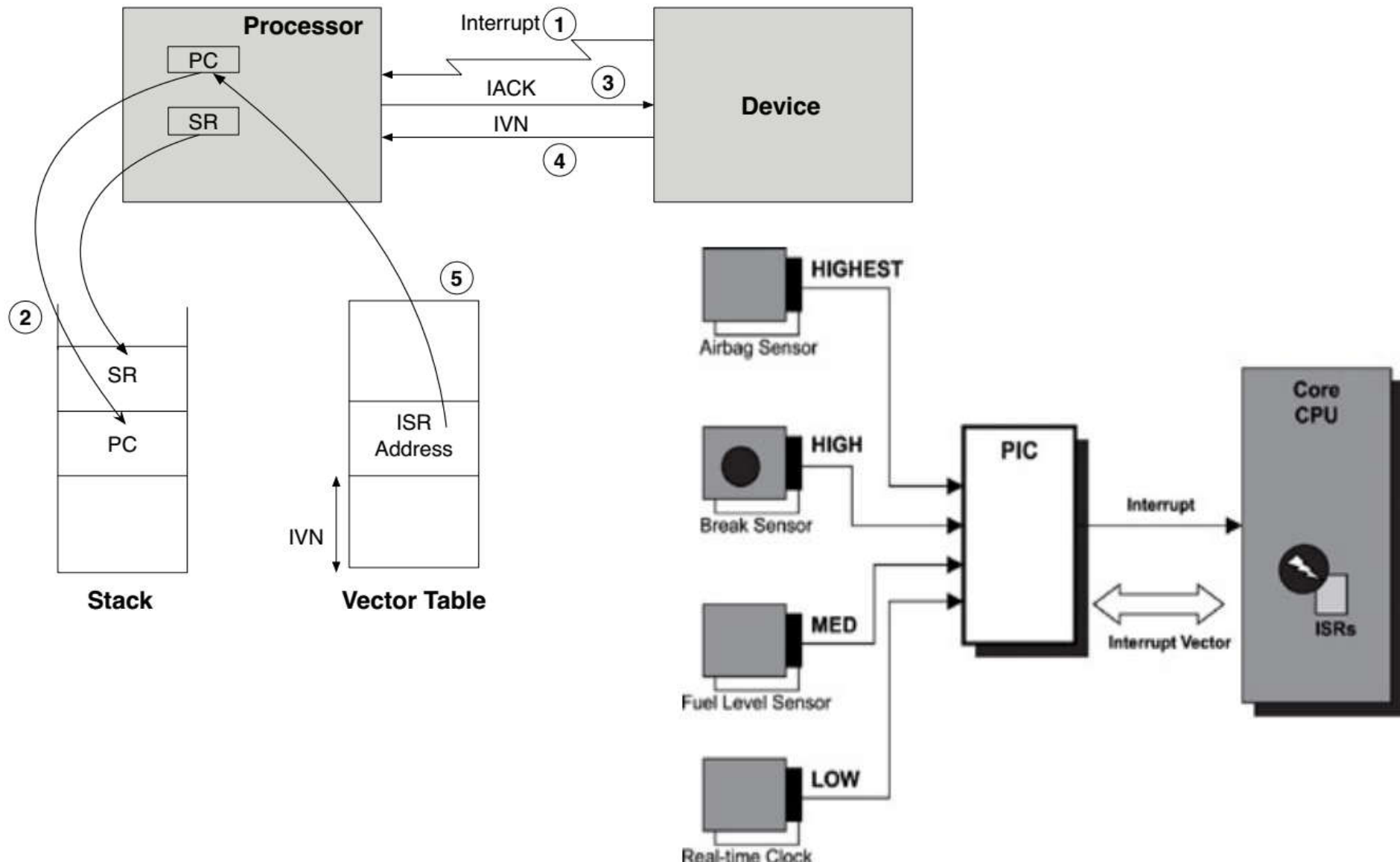


(b) Interrupt-Driven I/O



	No Interrupts	Use of Interrupts
I/O-to-memory transfer through processor	Programmed I/O	Interrupt-driven I/O
Direct I/O-to-memory transfer		Direct memory access (DMA)

The Interrupt Sequence

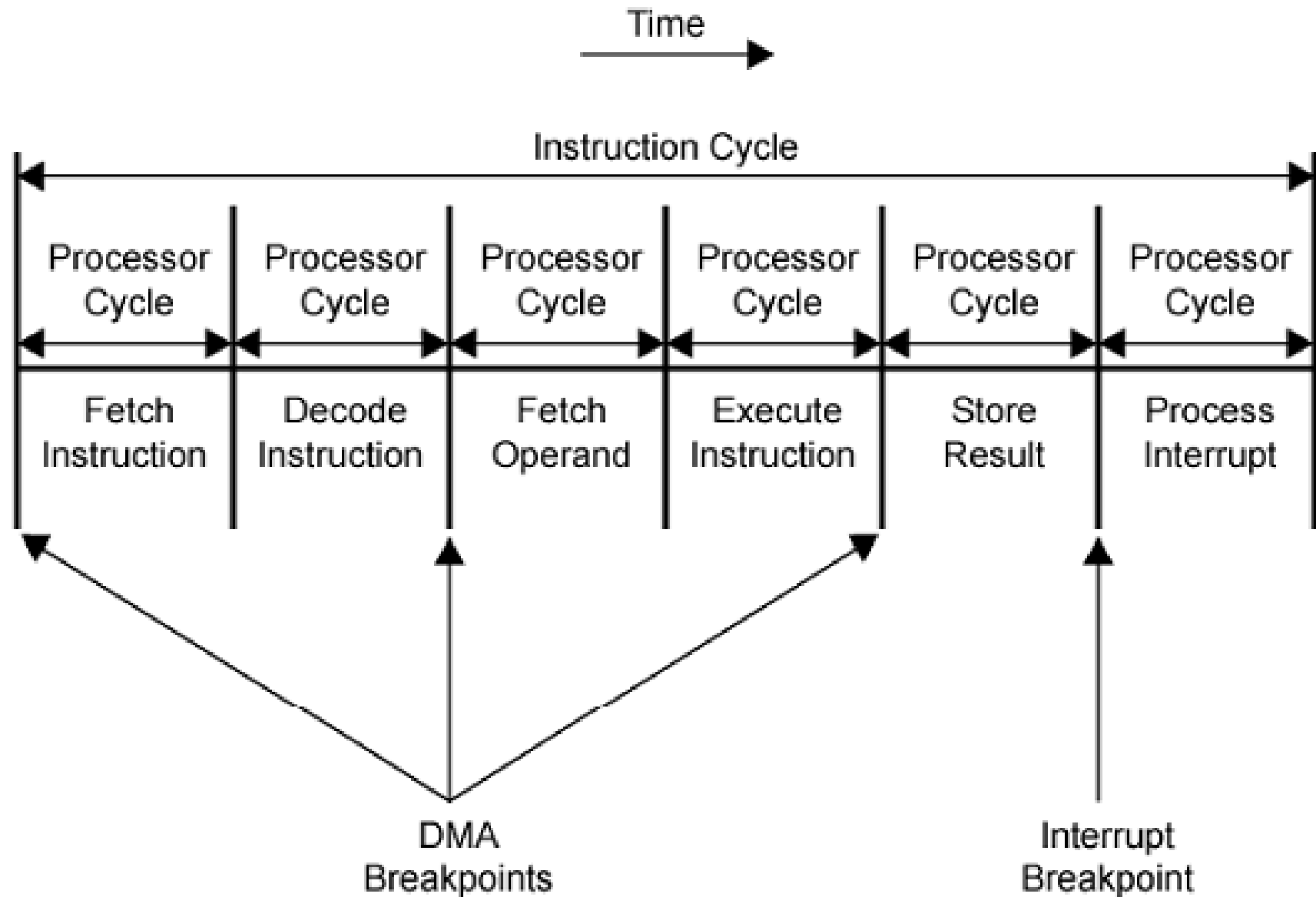




中断系统需解决的问题

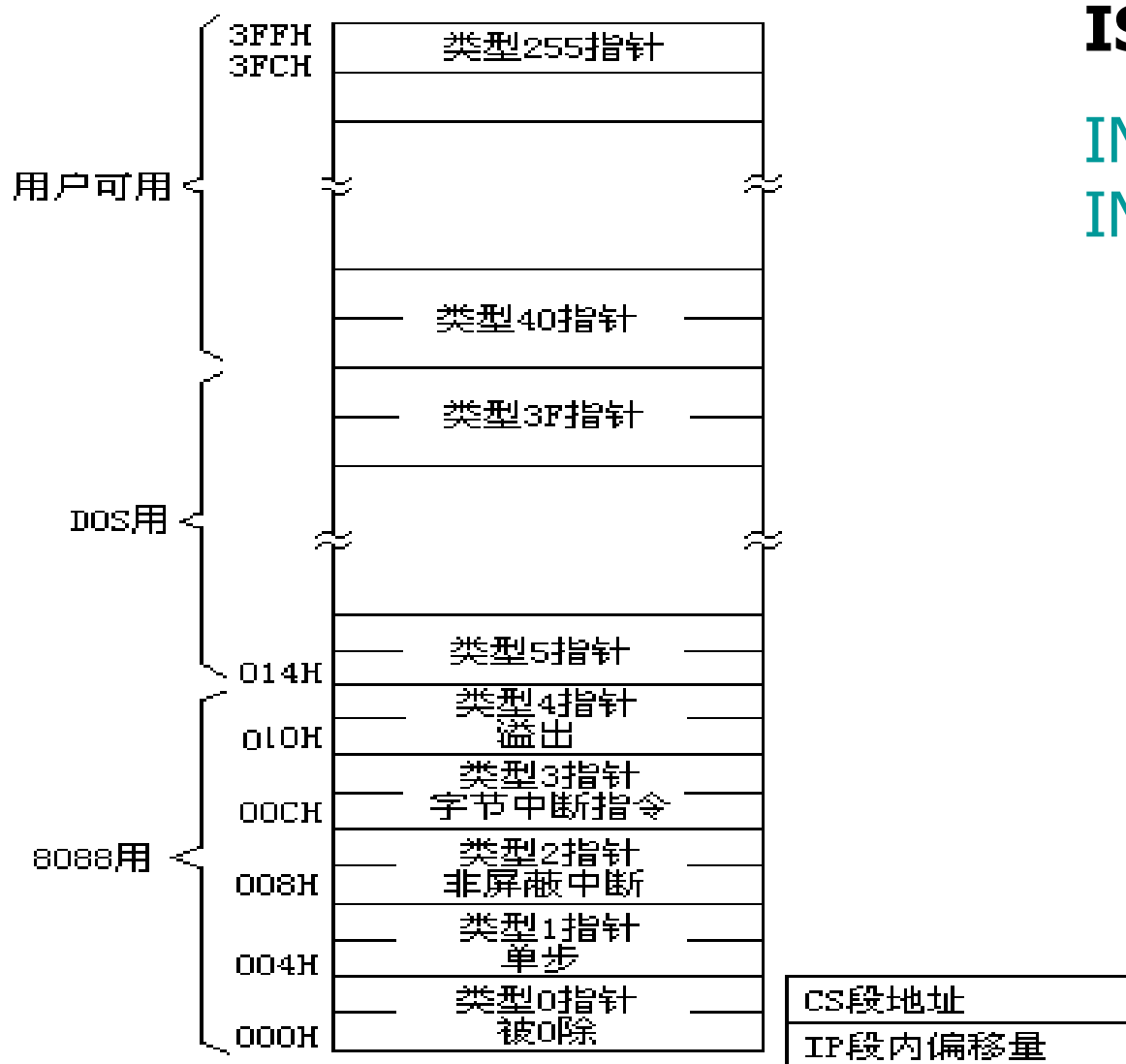
1. 如何确定中断源
 2. 出现多个中断时，中断响应的顺序
 3. 中断响应的条件、时机
 - 条件：处理器允许中断、该中断未被屏蔽
 - 时机：指令周期结束检查是否有中断请求
 4. 现场保护与恢复
 5. 中断服务程序的入口地址
 6. 中断处理过程中出现中断如何处理？
- 中断控制器

Interrupt and DMA Breakpoints





8086中断服务程序



中断向量表

ISR框架结构

INTHANDLE PROC FAR
INTHAND:

PUSH AX

PUSH BX

PUSH CX

...

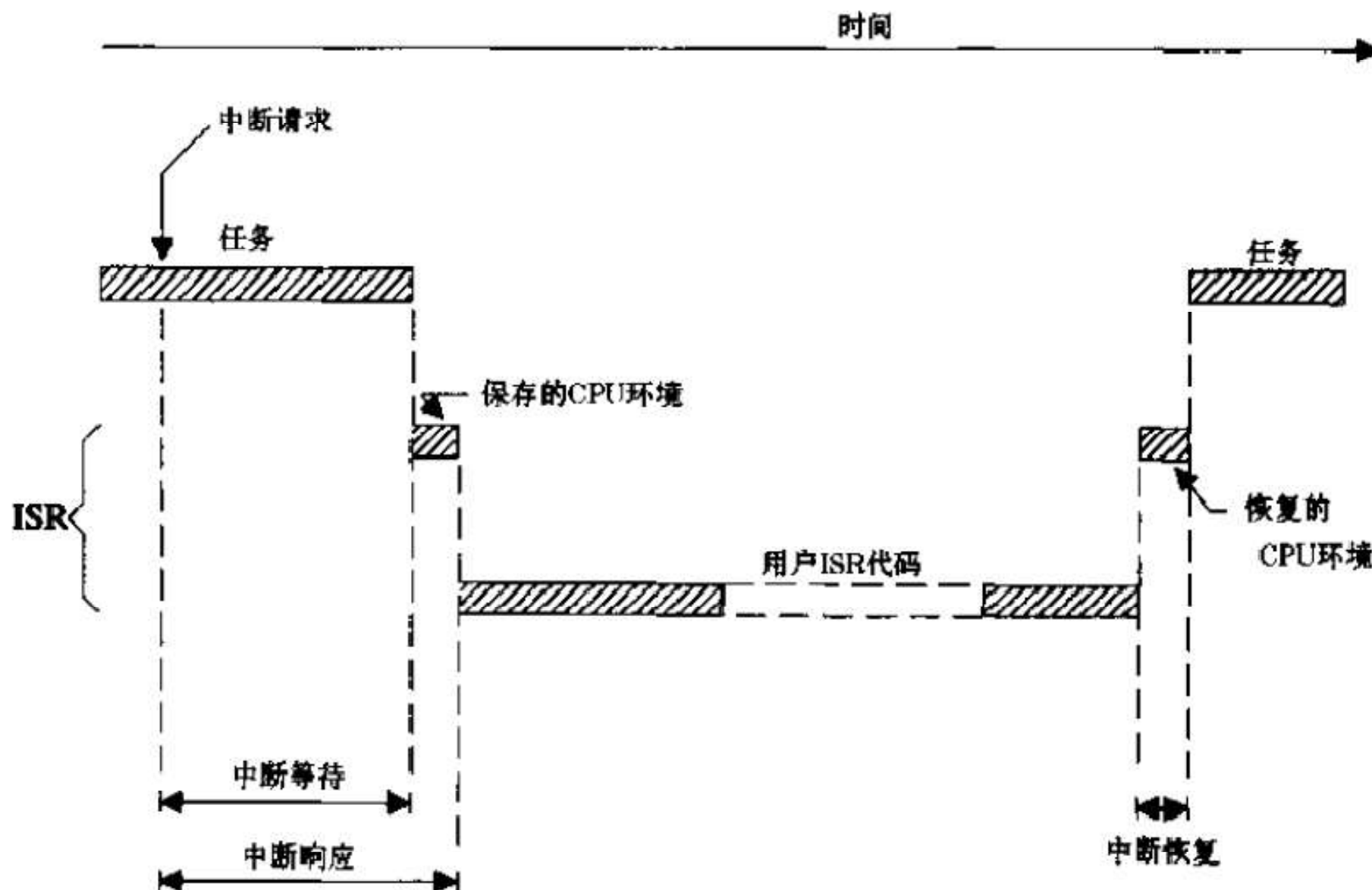
POP CX

POP BX

POP AX

IRET

中断响应时间：等待、服务、恢复



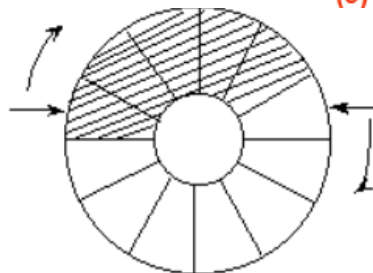
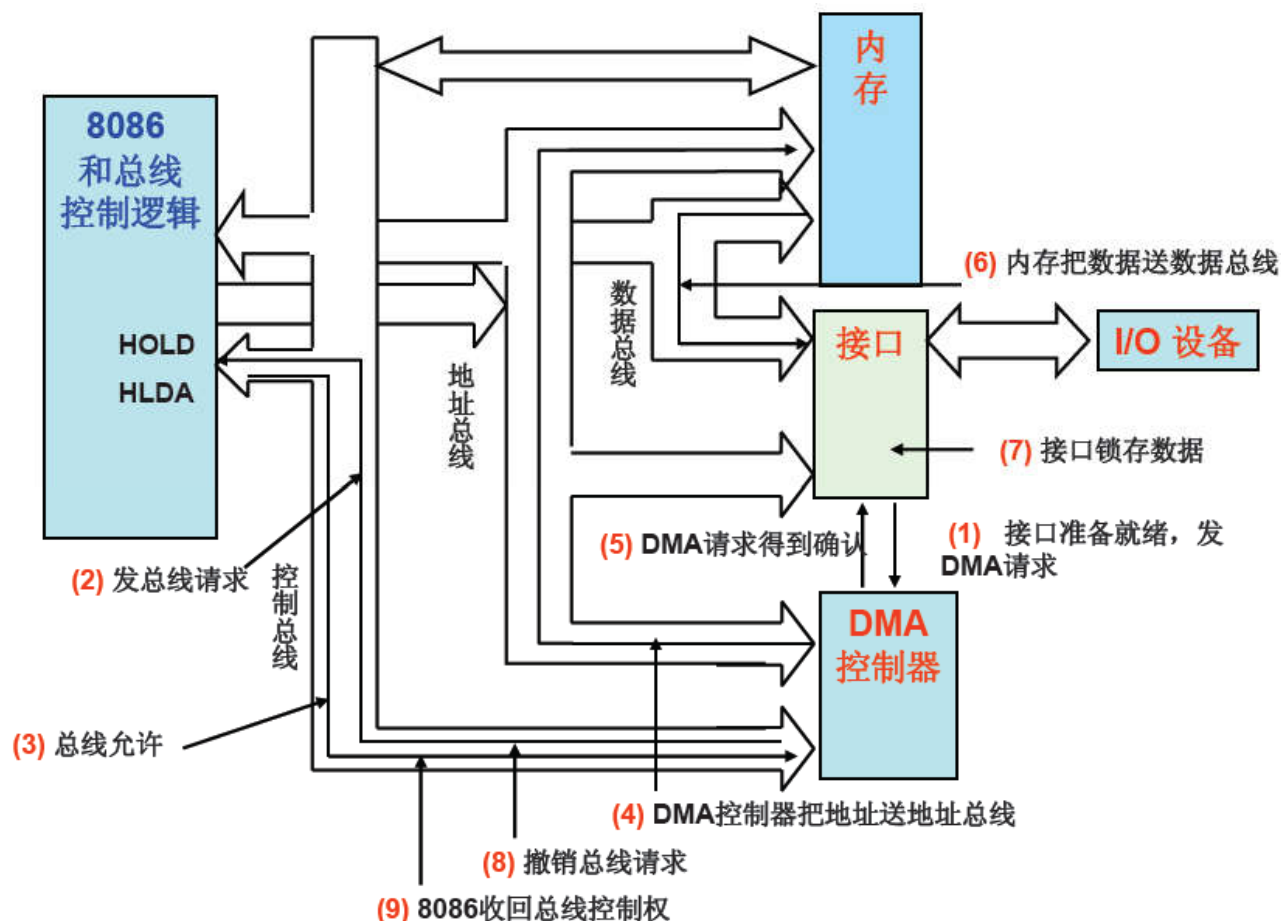
DMAC控制I/O



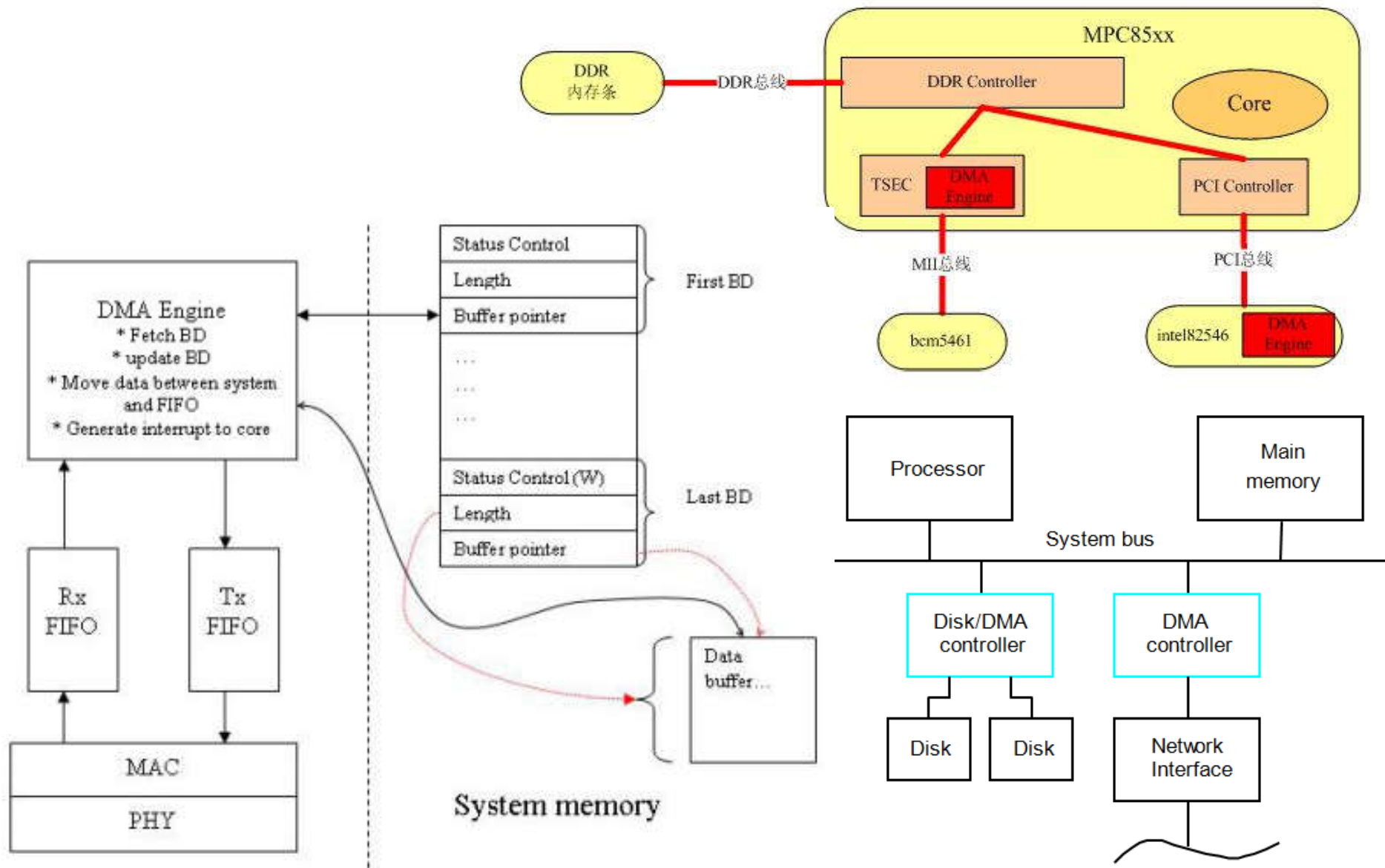
- 内存与外设
- 内存与内存
- 外设与外设

- 单字节
- 块传输
- 请求传输

- 环形缓冲区



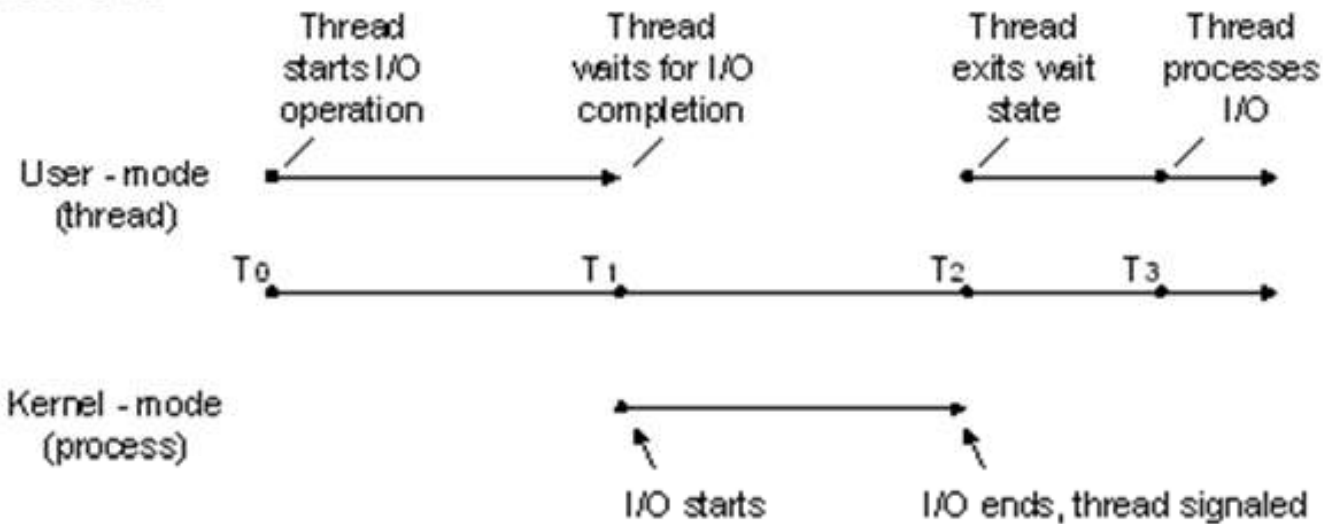
网络数据通信：网卡的DMA Engine



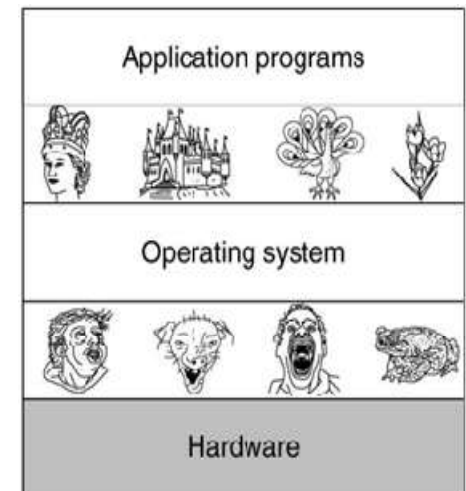
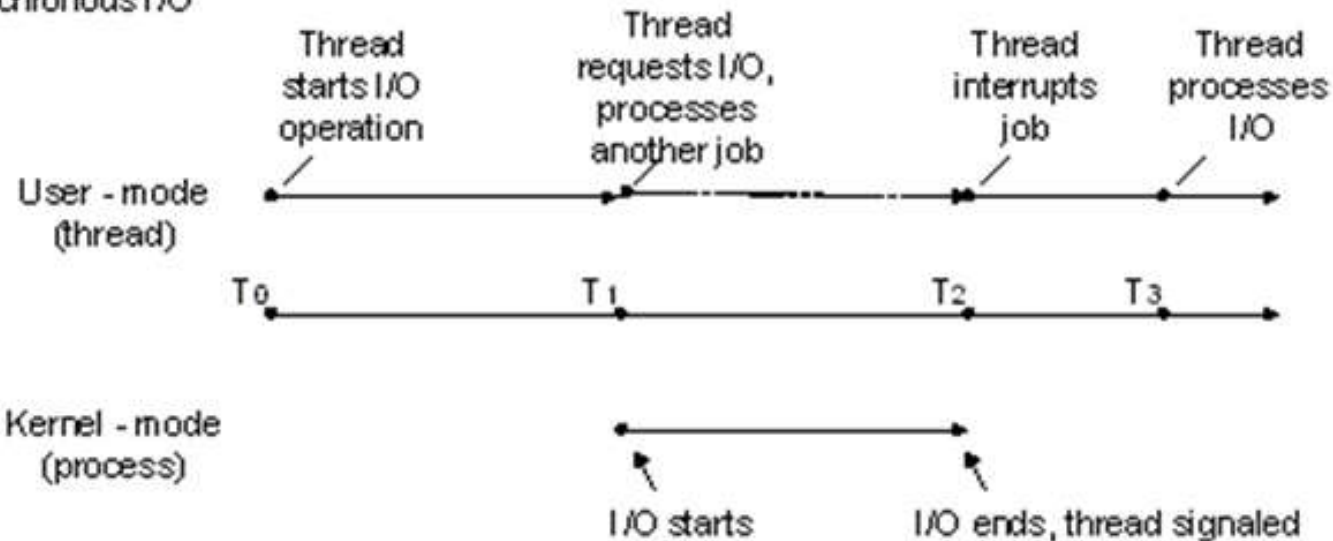
APP's two types of I/O Paradigms



Synchronous I/O



Asynchronous I/O



作业



- 影响流水线处理器性能发挥的主要因素有哪些？
- 从ILP和中断响应过程角度，分析中断响应时间jitter的产生原因及其范围。



Thank you