

Introduction on **RISC**

gsu@hust.edu.cn  
May17,2015

	CISC	RISC
价格	硬件复杂，芯片成本高	硬件较简单，芯片成本低
性能	减少代码尺寸，增加指令的执行周期数	使用流水线降低指令的执行周期数，增加代码尺寸
指令集	大量的混杂型指令集，有专用指令完成特殊功能	简单的单周期指令，不常用的功能由组合指令完成
应用范围	通用机	专用机
功耗与面积	含有丰富的电路单元，功能强、面积大、功耗大	处理器结构简单，面积小，功耗小
设计周期	长	短

**CPU, Go-Go-Go**

- **取指令**：当程序已在存储器中时，首先根据程序入口地址取出一条程序，为此要发出指令地址及控制信号。
- **分析指令**：即指令译码。是对当前取得的指令进行分析，指出它要求什么操作，并产生相应的操作控制命令
- **执行指令**：操作控制信号序列，通过运算器，存储器及输入/输出设备的执行，实现每条指令的功能，其中包括对运算结果的处理以及下一条指令地址的形成。

	CISC	RISC
内核结构	冯·诺依曼	哈弗
指令系统	不等长指令集，指令功能强	等长精简指令集，效率高
难点	实现复杂性	编译器复杂性
微处理器体系	微码	流水线
操作数	任意	寄存器中
典型用途	通用计算机	专用、嵌入式

**CPU的作用**

- 1) 能对指令进行译码并执行规定的动作
- 2) 可以进行算术和逻辑运算
- 3) 能与存储器/外设交换数据
- 4) 提供整个系统所需要的控制

**RISC**

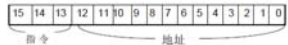

- Reduced Instruction Set Computer
- 复杂指令集计算机CISC(Complex Instruction Set Computer)

➤ 有限的简单的指令集

➤ 大量寄存器

➤ 指令流水线

用触发器和逻辑门直接连线构成的状态机和组合逻辑，产生控制序列的速度要快于微程序控制

## RISC的特点及设计思想

RISC机的设计应当遵循以下五个原则：

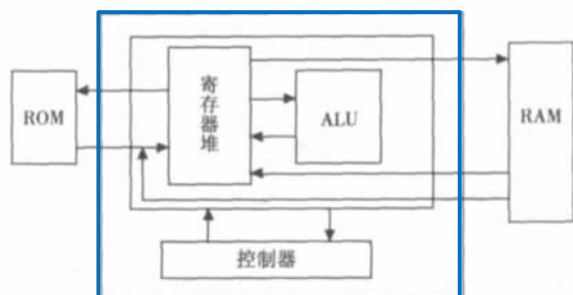
- ① 指令条数少，格式简单，易于译码；
- ② 提供足够的寄存器，只允许load和store指令访问内存；
- ③ 指令由硬件直接执行，在单个周期内完成；
- ④ 充分利用流水线；
- ⑤ 强调优化编译器的作用；

## 最简的CPU内核

- 时钟发生器：分频生成系列时钟信号
- 指令寄存器：指令存入寄存器
- 累加器：存放当前的运算结果
- 算术逻辑运算单元：加、与、异或、跳转
- 数据控制器：控制累加器的数据输出
- 状态控制器、程序控制器：有限状态机
- 程序计数器：提供指令地址
- 地址多路器：程序计数器地址/数据/端口地址

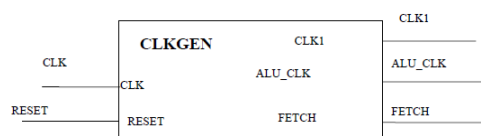
10

## A Simple CPU



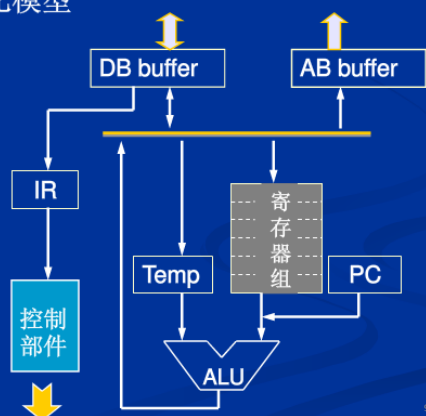
8

## 时钟就是产生相位



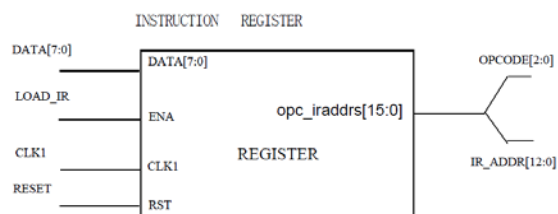
11

## CPU简化模型



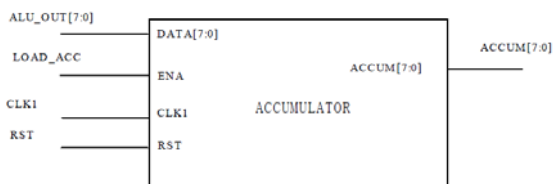
9

## 指令寄存器



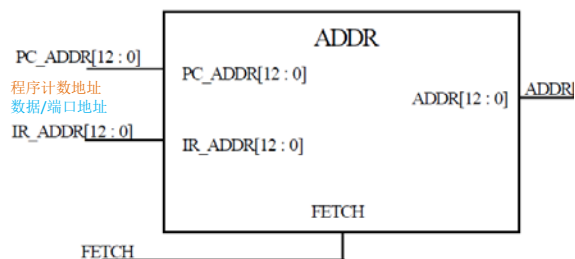
12

累加器其实是双目运算的暂存



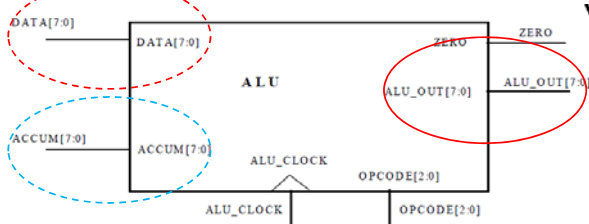
13

地址多路器就是二选一多路器



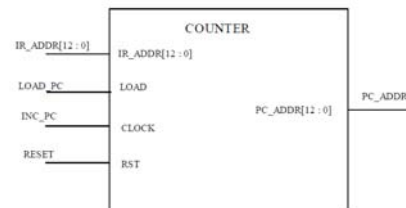
16

ALU就是按OPCODE分若干case运算



14

程序计数器就是个计数器

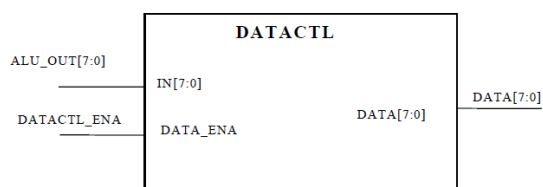


```

if(rst)
    pc_addr<=13'b0_0000_0000_0000; 复位
else if(load)
    pc_addr<=ir_addr; 跳转执行
else
    pc_addr <= pc_addr + 1; 顺序执行
    
```

17

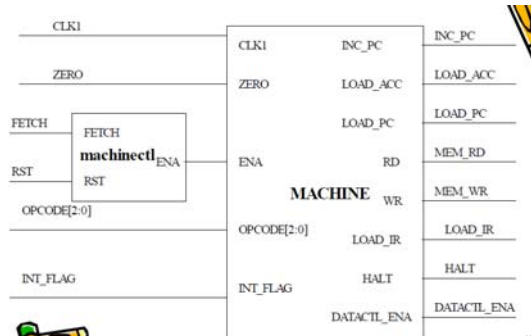
数据控制器，若干三态门而已



高阻，高祖

15

状态控制真的很复杂？！



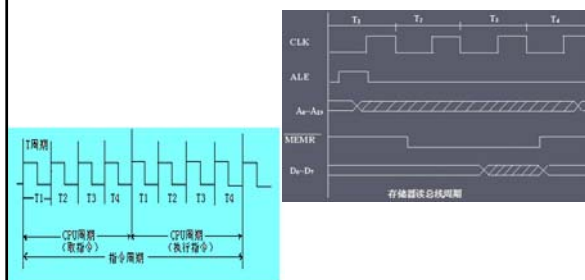
18

## RISC状态机其实很简单

- 严格时钟节拍，每个节拍做固定的任务  
取指H、取指L、空、PC+、读数、写数、空、PC+
- 所谓固定的任务：按OPCODE写case语句
- 复位就是全部清零

19

## 读MEM



22

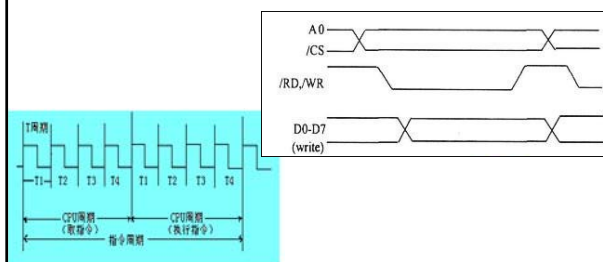
## 仅有CPU，仅能发热



- 存储测试程序的ROM
- 装载数据的RAM
- 地址译码器

20

## 写MEM，逻辑时序和读一样



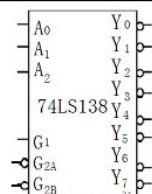
23

## 地址译码器产生选通信号

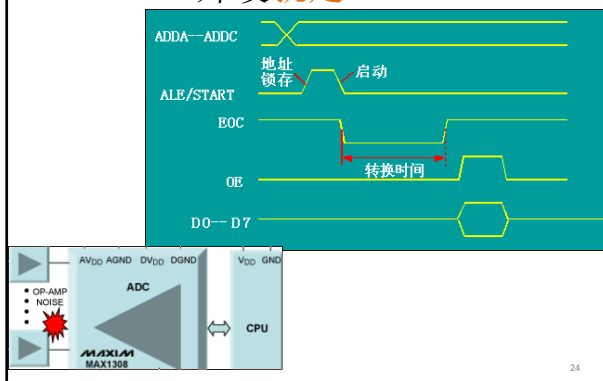
```

case (addr)
  13'b1-1xxx-xxxx-xxxx: {rom_sel, ram_sel} <= 2'b01;
  13'b0-xxxx-xxxx-xxxx: {rom_sel, ram_sel} <= 2'b10;
  13'b1-0xxx-xxxx-xxxx: {rom_sel, ram_sel} <= 2'b10;
  default: {rom_sel, ram_sel} <= 2'b00;
endcase

```



## 外设就是MEM



24

## 关键技术：流水线技术



多个功能部件并行工作来缩短程序执行时间

25

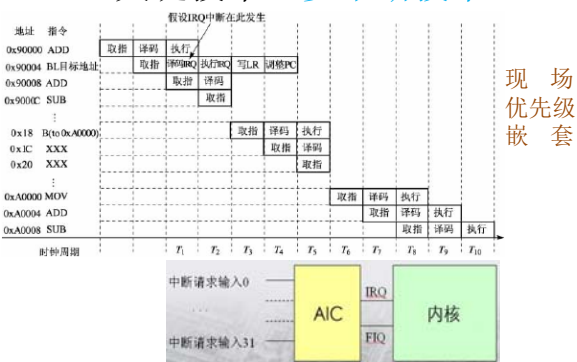
## 关键技术：分页设计

- 具有共享区的寄存器堆的分页设计
- 程序空间的分页设计

分段、分页的设计出发点：  
将存储器的线性地址分解成二维或多维地址

28

## 关键技术：多中断技术



26

## 关键技术：低功耗设计

- 优化指令集，简化系统的译码单元和执行单元
- 硬件并行以及功能单元的流水执行实现低功耗结构
- 确定存储器、寄存器的容量，减少总线数目
- 硬件的各个子模块划分、软件设置不同工作状态

29

## 关键技术：指令集的选取

- 权衡指令的长度
- 指令字节格式
- 寄存器数量
- 指令字节格式分配
- 存储器、寄存器、I/O口是否统一寻址

27

