计算机系统结构

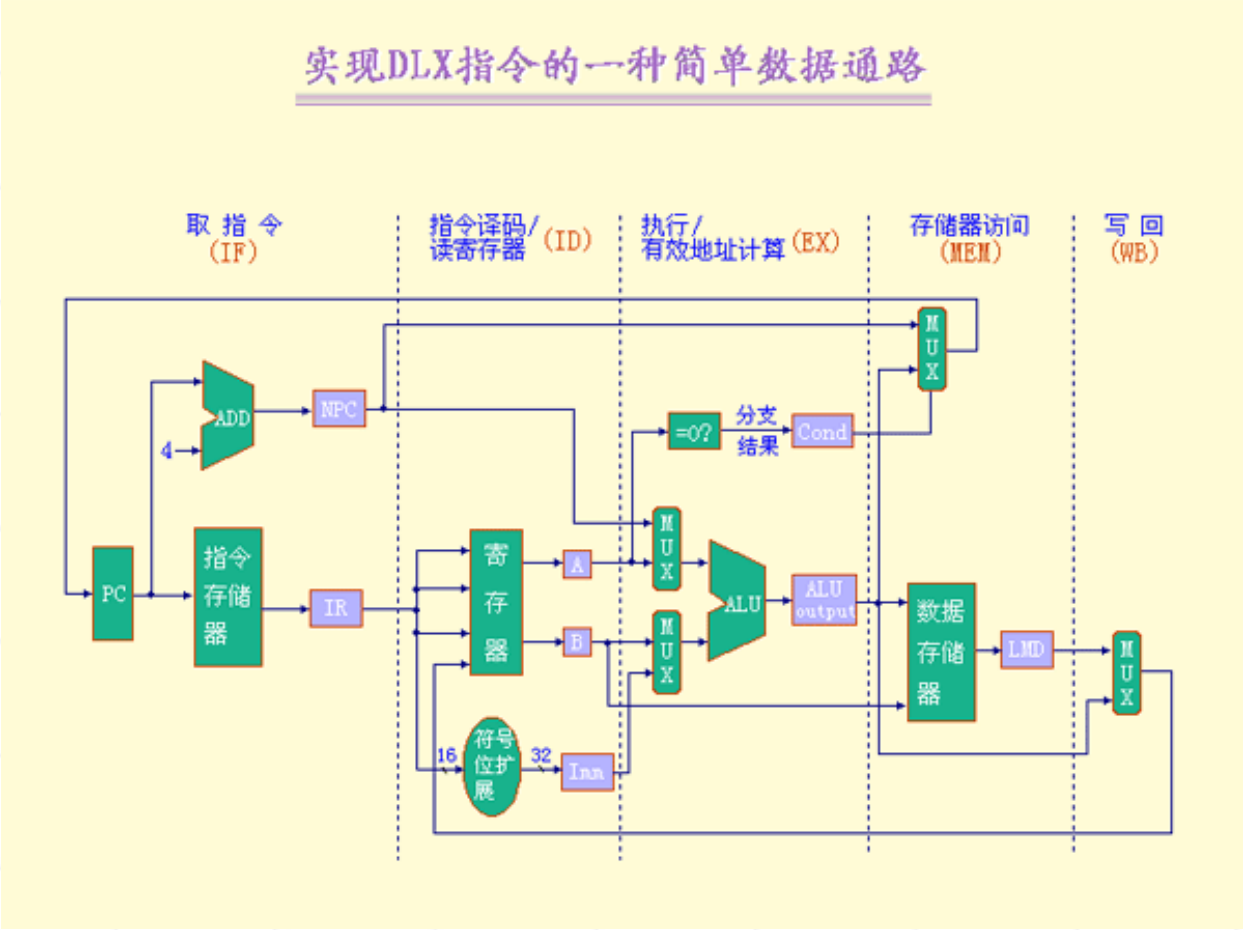
第八讲流水线的实现（1）

冯丹

武汉光电国家研究中心

华中科技大学
HUAZHONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

1-1. MIPS（子集）数据通路（非流水）实例



1-2.取指令周期（IF）

- 操作
  - IR←Mem[PC]
  - NPC←PC+4

1-3.指令译码/读寄存器周期（ID）

- 主要操作
  - A ← Regs[rs]
  - B ← Regs[rt]
  - 指令译码
  - Imm ← （（IR16）16##IR16..31）
- 备注
  - 指令的译码操作和读寄存器操作是并行进行的：在MIPS指令格式中，操作码字段以及rs、rt字段都是在固定的位置。这种技术称为固定字段译码技术。

1-4.执行/有效地址计算周期（EX）

- 主要操作（不同指令进行的操作不同）
  - load指令和store指令
    - ALUo←A + Imm
  - 寄存器－寄存器ALU指令
    - ALUo←A funct B
  - 寄存器－立即值ALU指令
    - ALUo←A op Imm
  - 分支指令
    - ALUo←NPC+（Imm<<2）
    - cond←（A = = 0）
- 备注
  - 将有效地址计算周期和执行周期合并为一个时钟周期，这是因为MIPS指令集采用load / store结构，没有任何指令需要同时进行数据有效地址的计算、转移目标地址的计算和对数据进行运算。

1-5.存储器访问/分支完成周期（MEM）

– 主要操作（不同指令进行的操作不同）

- 所有指令都要在该周期对PC进行更新
  - 非分支指令：PC←NPC
  - 分支指令：if （cond） PC ←ALUo else PC←NPC
- 该周期内需要处理的MIPS指令仅包含load、store和分支三种
  - load指令和store指令：LMD←Mem[ALUo]或者 Mem[ALUo]←B
  - 分支指令：如上所述，更新PC

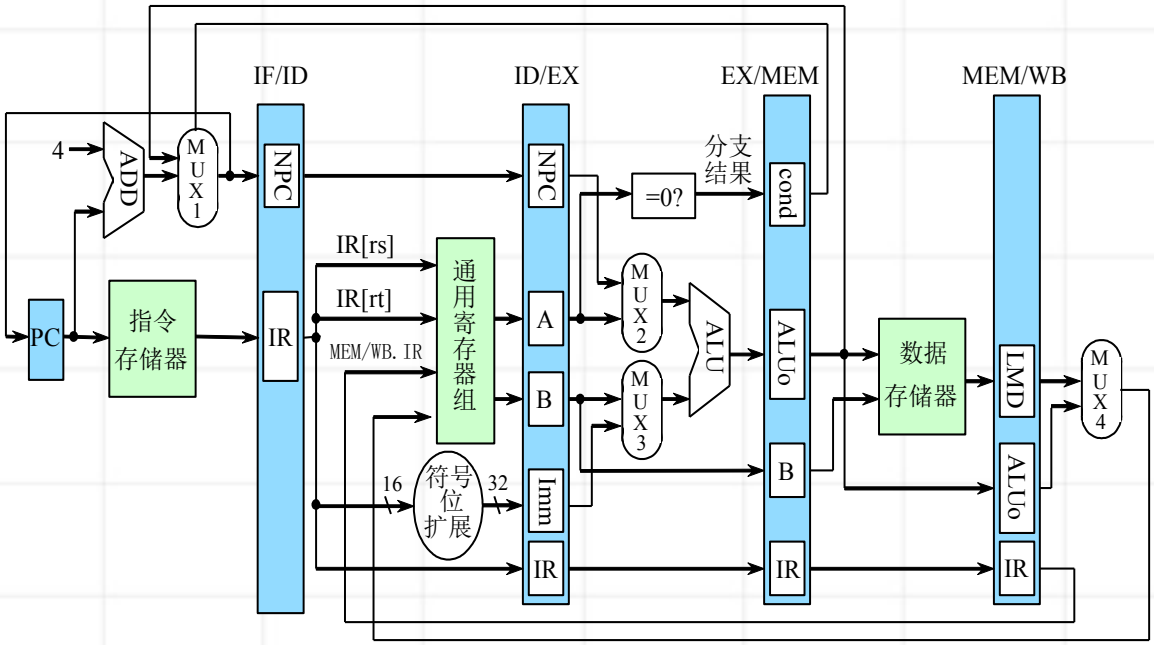
1-6.写回周期（WB）

- 主要操作（不同指令进行的操作不同）
  - 寄存器－寄存器ALU指令
    - Regs[rd]← ALUo
  - 寄存器－立即数ALU指令
    - Regs[rt]← ALUo
  - load指令
    - Regs[rt]← LMD

2.MIPS的流水化

– 基本原则

- 每一个时钟周期完成的工作看作是流水线的一段，每个时钟周期启动一条新的指令。



流水实现的数据通路

3-1.流水寄存器（1）位置与作用

- 位置
  - 段与段之间设置流水寄存器
- 作用
  - 将各段的工作隔开，使得它们不会互相干扰。
  - 保存相应段的处理结果。
    - 例：EX/MEM.ALUo：保存EX段ALU的运算结果
  - 向后传递后面将要用到的数据或者控制信息
    - 所有有用的数据和控制信息每个时钟周期会随着指令在流水线中的流动往后流动一段

3-2.流水寄存器（2）命名规则

- 流水寄存器名.子寄存器名[字段名]
  - 流水寄存器名
    - 用其相邻的两个段的名称拼合而成，例如：ID段与EX段之间的流水寄存器用ID/EX表示
    - 每个流水寄存器是由若干个子寄存器构成的
  - 子寄存器名
    - 基本寄存器，如IR
  - 字段名
    - 基本寄存器的某字段
  - 例子
    - ID/EX.IR[op]：流水寄存器ID/EX中的子寄存器IR的op字段（即操作码字段）

4.其它数据通路改动

- 增加了向后传递IR和从MEM/WB.IR回送到通用寄存器组的连接。
- 将对PC的修改移到了IF段，以便PC能及时地加4，为取下一条指令做好准备。

5-1.通用操作

- 操作
  - IR[rs]＝$IR_{6..10}$
  - IR[rt]＝$IR_{11..15}$
  - IR[rd]＝$IR_{16..20}$

5-2.取指令周期（IF）

- 操作
  - IF/ID.IR ← Mem[PC]
  - IF/ID.NPC, PC ← （if（（ EX/MEM.IR[op] == branch ）& EX/MEM.cond）{EX/MEM.ALUo} else {PC+4}）；

5-3.指令译码/读寄存器周期（ID）

- 主要操作
  - ID/EX.A ← Regs[IF/ID.IR[rs]]
  - ID/EX.B ← Regs[IF/ID.IR[rt]]
  - ID/EX.NPC ← IF/ID.NPC；
  - ID/EX.IR ←IF/ID.IR；
  - ID/EX.Imm ← （IF/ID.IR16)16##IF/ID.IR16..31；

5-4.执行/有效地址计算周期（EX）

- 主要操作（不同指令进行的操作不同）
  - 所有指令共同操作
    - EX/MEM.IR ← ID/EX.IR;
  - load指令和store指令
    - EX/MEM.ALUo ←ID/EX.A + ID/EX.Imm;
    - EX/MEM.B←ID/EX.B;
  - 寄存器－寄存器ALU指令
    - EX/MEM.ALUo ←ID/EX.A funct ID/EX.B
  - 寄存器－立即值ALU指令
    - EX/MEM.ALUo ← ID/EX.A op ID/EX.Imm
  - 分支指令
    - EX/MEM.ALUo ←ID/EX.NPC+ID/EX.Imm<<2;
    - EX/MEM.cond ← （ID/EX.A ==0）;

5-5.存储器访问/分支完成周期（MEM）

- 主要操作（不同指令进行的操作不同）
  - ALU与L/S指令共同操作
    - MEM/WB.IR←EX/MEM.IR
  - ALU指令（两类：寄存器与寄存器，以及寄存器——立即数）
    - MEM/WB.ALUo←EX/MEM.ALUo
  - load指令和store指令
    - MEM/WB.LMD ←Mem[EX/MEM.ALUo]
    - Mem[EX/MEM.ALUo] ←EX/MEM.B;

5-6.写回周期（WB）

- 主要操作（不同指令进行的操作不同）
  - 寄存器－寄存器ALU指令
    - Regs[MEM/WB.IR[rd]] ←MEM/WB.ALUo
  - 寄存器－立即数ALU指令
    - Regs[MEM/WB.IR[rt]] ←MEM/WB.ALUo
  - load指令
    - Regs[MEM/WB.IR[rt]] ←MEM/WB.LMD

谢谢大家