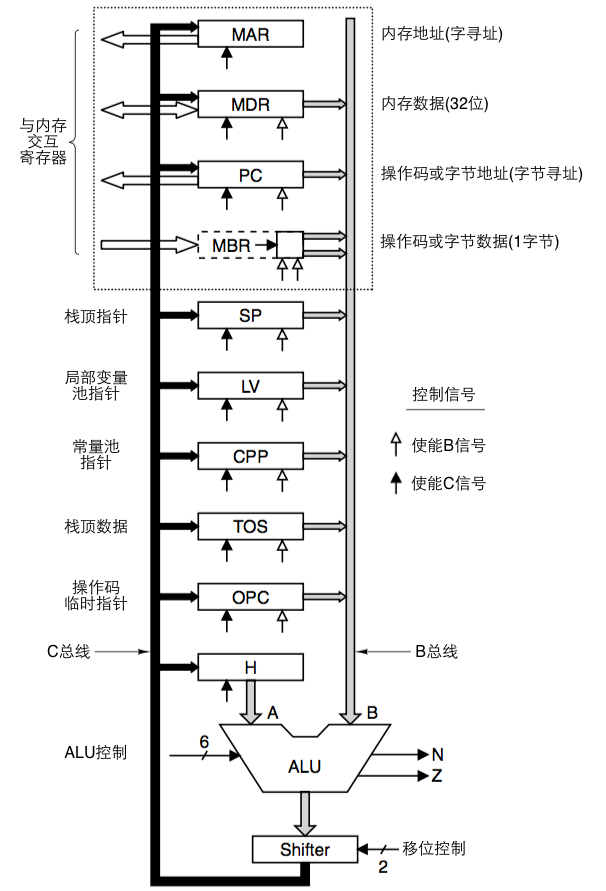
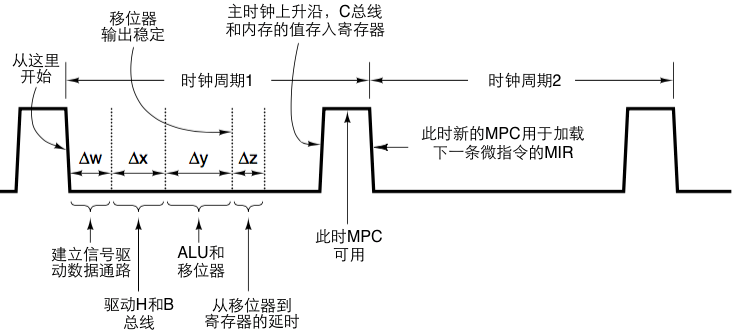
**一、微体系的概念**

**1数据通路**

概念：数据存入寄存器经由ALU运算后再次存入寄存器的过程



数据通路时序：数据通路周期中各数字逻辑的机器周期



**2内存操作**

内存操作寄存器

MAR (只写)内存地址寄存器，指定读取的内存字地址

MDR 内存数据寄存器

PC 程序计数器，改变计数器将实现程序的跳转

MBR (只读)内存字节数据寄存器，获取微指令或字节数据

内存的操作过程：若当前周期K有内存操作信号

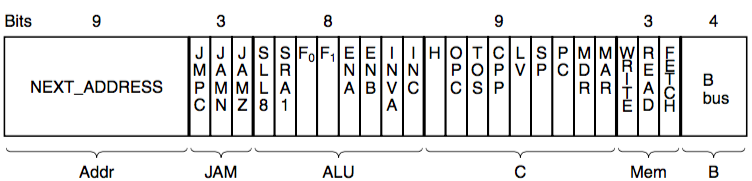
·K+1周期开始时启动内存操作(第一周期结束上升沿MAR、PC被加载)

·K+1周期结束上升沿，内存被存入寄存器MDR、MBR

·K+2周期可用MDR或MBR

**3微指令**

微指令：提供数据通路和内存操作的控制信号



Addr 下一条微指令地址

JAM 微指令跳转控制

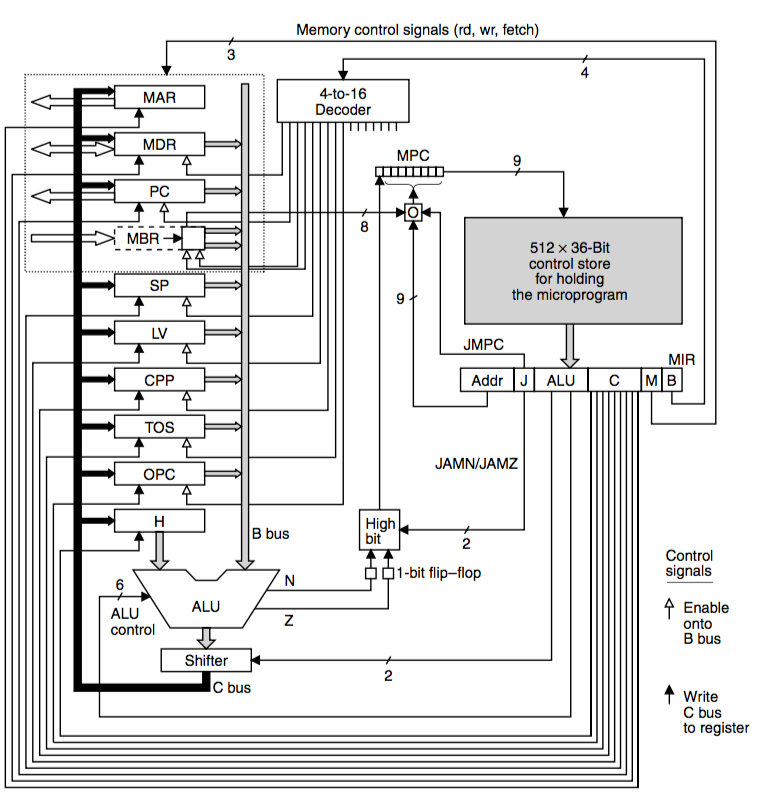
ALU ALU和移位器控制信号

C C总线控制信号

Mem 内存控制信号

B B总线控制信号

微指令控制：决定微指令的执行顺序

****

control store 存放全部微程序的ROM

MPC 微程序计数器

MIR 微指令寄存器

**二、三总线体系**

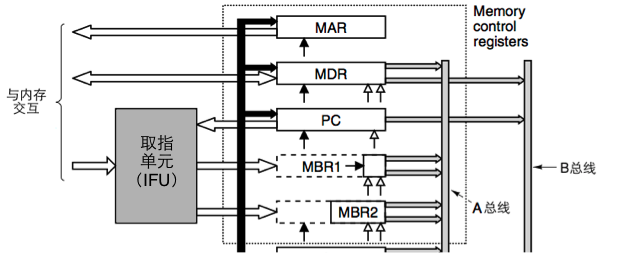
设计：ALU配备两条完成的输入总线，两条输入总线分别连接所有寄存器

作用：可以在一个周期内执行任意两个寄存器的加法

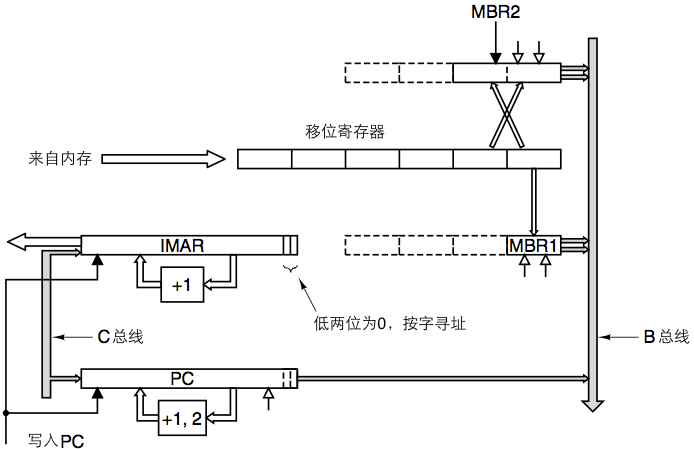
**三、取指单元(IFU)**

使主执行单元直接获取指令，无需等待内存操作

**1设计**



**2 IFU结构**



·IMAR 内存地址寄存器，按字寻址取字数据

·移位寄存 预取数据的缓冲区，读MBR1/2结束时将触发移位操作

·MBR1/2 分别保存1字节、2自己的指令或数据

**3相关操作**

读取MBR1/2：读完成时移位寄存器移位，并用新数据填出寄存器

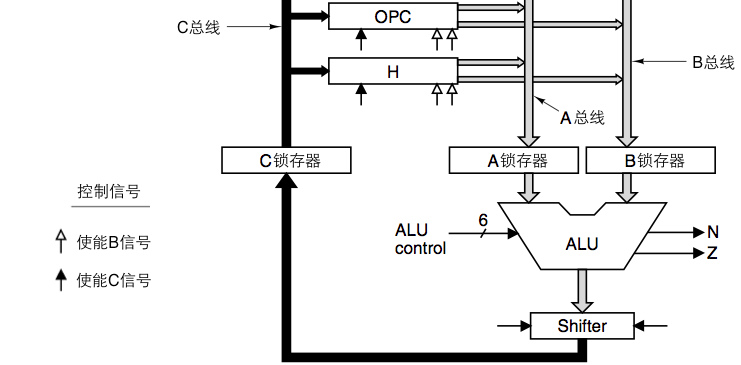
写入PC(程序跳转)：PC写入时，移位寄存器被清空，IMAR将被重置

预取指令操作：根据移位寄存器状态，IMAR自增方式预取字数据

**四、流水线执行**

可工作在更高频率下(步骤少延迟降低了)；提高了指令吞吐率(每个步骤均被使用)

**1设计(增加3个锁存器)**



**2流水线的执行**

执行步骤

·第一个周期锁存器A和B加载数据

·第二个周期ALU和移位寄存器执行运算并存入锁存器C

·第三个周期把锁存器C写回寄存器中

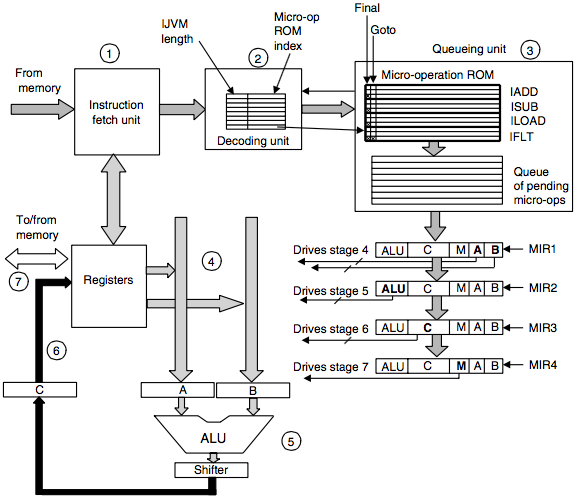
执行相关性：流水线遇到相关性的寄存器时，将不得不等待执行

·RAW相关 读取等待写入，某指令需要前一条指令的计算结果

·WAR相关 写入等待读取，某保存结果的寄存器正被前一条指令读取

·WAW相关 写入等待写入，某保存结果的寄存器正被钱一条指令写入

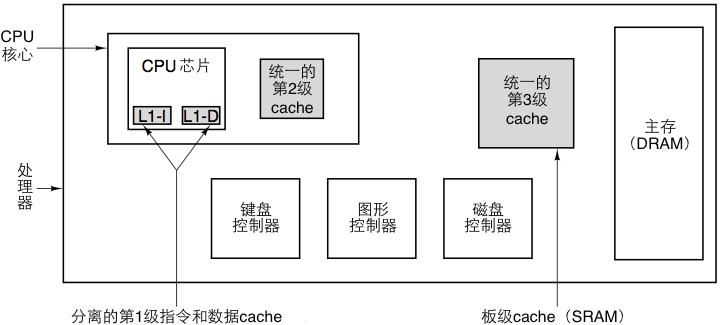
**3七段流水线设计**



**五、高速缓存**

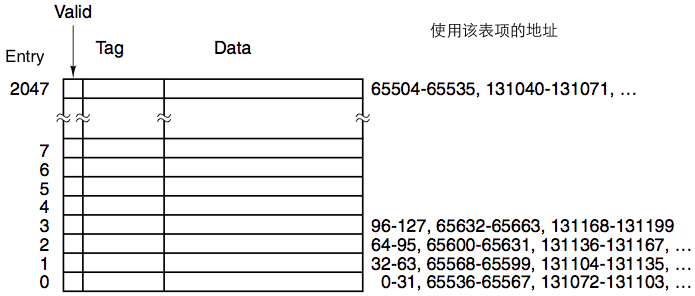
减少对内存的访问，以降低存储器延时

**1设计(多级cache存储)**



**2缓存模型结构**

主存被成分块后映射到高速缓存cache中



·Entry 表项，此cache被分成2048项

·Valid 有效位，标示当前cache项数据是否有效

·Tag 标示cache属于主存中的哪个数据块

·Date 数据字段，这里为32字节

**3存取cache操作**

存取cache的地址模型



·TAG 指定cache模型中Tag字段，指定主存对应的数据块

·LINE 指定cache模型中Entry字段，指定cache的项

·Word 指定cache项中的某个字(一个cache项有8个字)

·BYTE 指定Word对应字中的某个字节

读取的操作

·用LINE匹配cache的Entry，用TAG匹配cache的Tag

·若命中(均匹配且有效)，直接存取caches数据

·若未命中，先取内存替换该cache项，再存取

cache项已被修改的要先写回内存

**4组相连高速缓存**



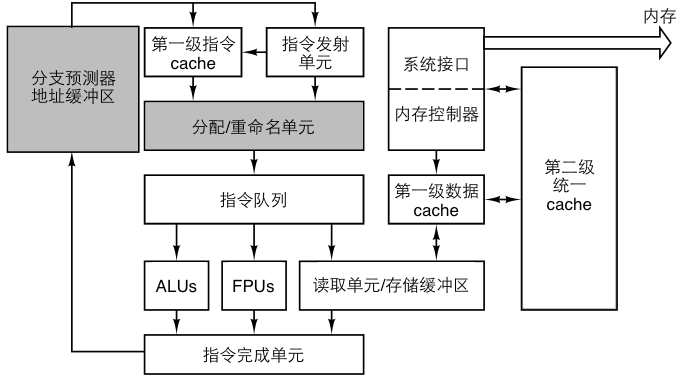
普通cache的问题：内存中不同块会竞争相同cache

组相连cache作用：1个cache可映射多个内存快，解决了竞争cache的问题

**六、分支预测**

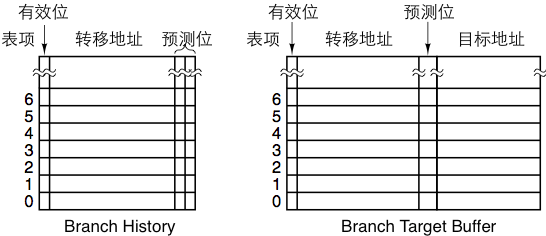
**1设计**

解决分支指令导致流水线阻塞问题



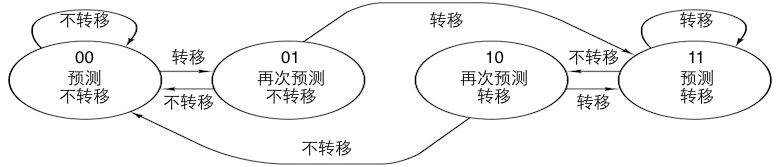
**2动态分支预测**

根据分支历史预测：CPU记录历史发生的条件转移指令，并以此作为再次发生的预测



·转移地址 分支指令出现的地址

·预测位 2位有限状态机



·目标地址 BTB提供转移的目的地址(如目的地址为寄存器寻址的)

两级自适应预测：略

**3静态分支预测**

由编译器设置指令中的一位，取指单元将按照编译器来采取行动

**4分支预测的处理过程(动态预测)**

遍历BTB：取指阶段用PC寻址BTB

·若命中，则使用BTB提供的分支地址取指

·未命中，则进入下述环节

遍历BHT：BTB的空间有限，若BTB未命中则遍历BHT

·若命中，则使用BHT的预测位

·未命中，则使用静态预测

暂存预测执行结果：若预测修改寄存器，则用别名寄存器保存临时结果

·若预测与结果吻合，则重命名寄存器

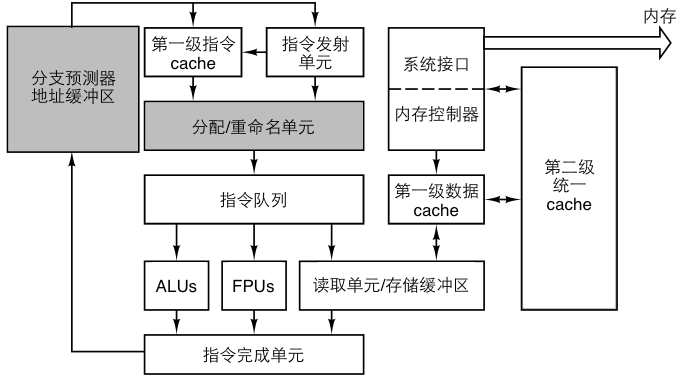
·若预测与结果不吻合，则丢弃重命名的寄存器

将条件转移结果写回BTB和BHT：根据预测结果选择性的写回

**七、乱序与寄存器重命名**

避免流水线的相关性执行(RAW、WAR、WAW)

**1设计**



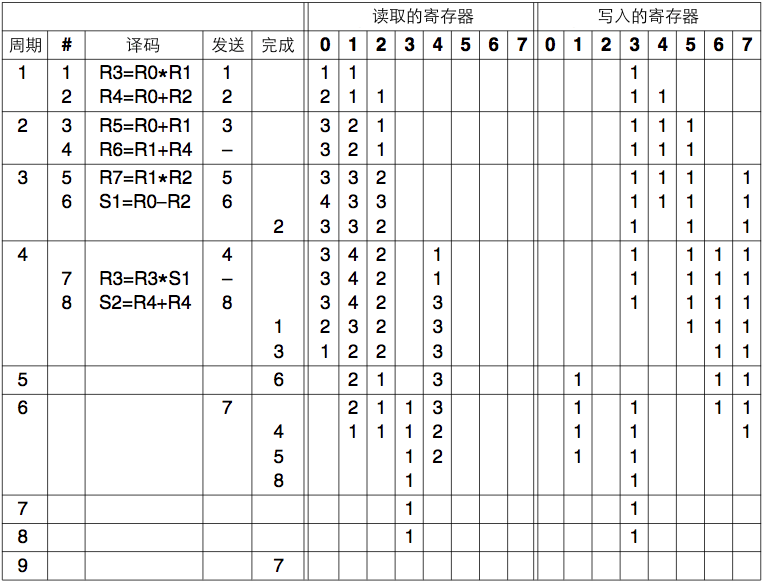
超标量CPU：一周期可启动多条(4~6条)指令给不同功能单元执行(ALU、FPU等单元)

基本块：一系列顺序执行的指令，不包含任何控制结构(if、while等)

适用范围：乱序重命名主要用在一个基本块内的指令

**2分配和重命名单元结构**

记分牌模型：描述寄存器使用情况，分配单元、重命名单元都会用到



分配单元：利用记分牌将指令乱序发送给功能单元执行，避免RAW相关

重命名单元：利用记分牌将指令的寄存器重命名，避免WAR、WAW相关

**八、推测执行**

在不知道是否需要执行之前就执行代码