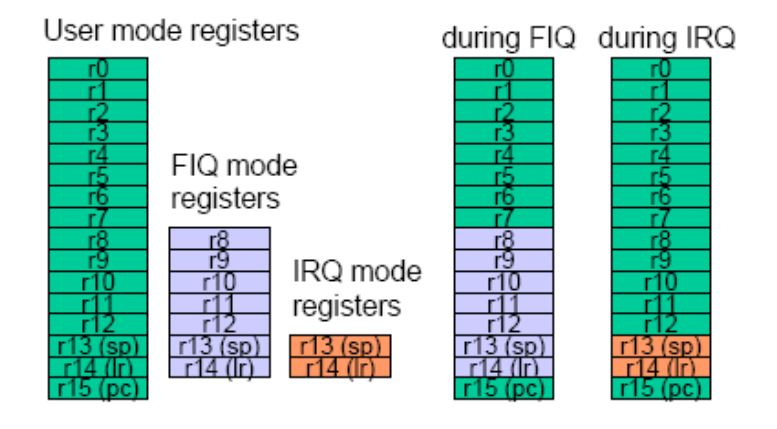
**Interrupt**

当一个interrupt发生时，ARM7会改变运行状态或者模式

ARM7有两种Interrupt模式

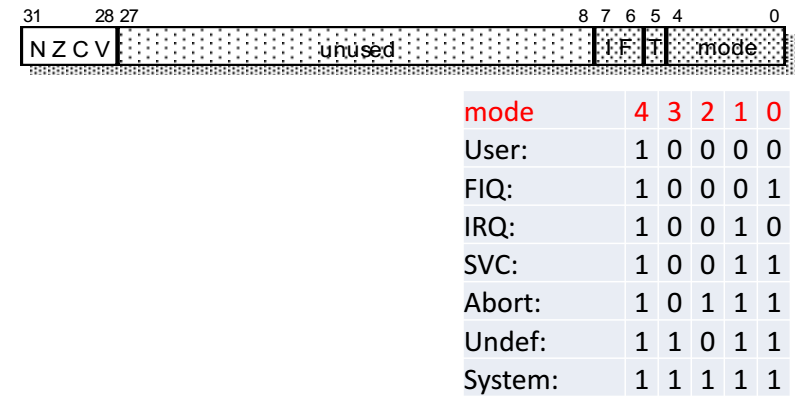
normal interrupt(IRQ)

fast interrupt(FIQ)



**Current Program Status Register**

CPSR是user-mode用来状态码bit的



**Interupt Handler**

当interrupt发生时，

IRQ或FIQ状态的register被激活

CPSR被储存到saved program status register（SPSR）中

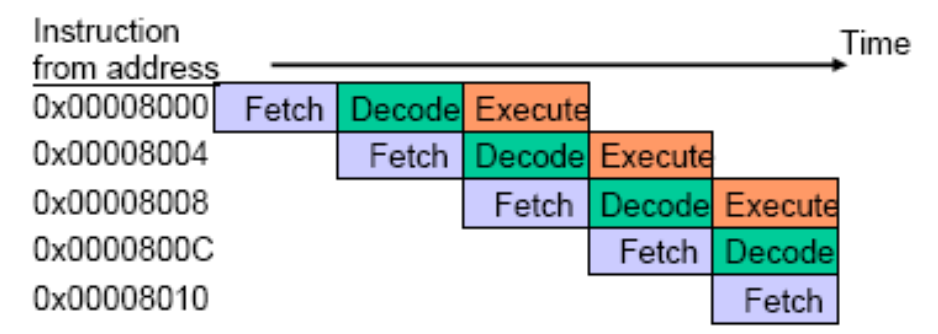
返回的指令地址被储存到相应的link register里。

The program counter is set to either 0x00000018 for an IRQ or 0x0000001C for a FIQ.

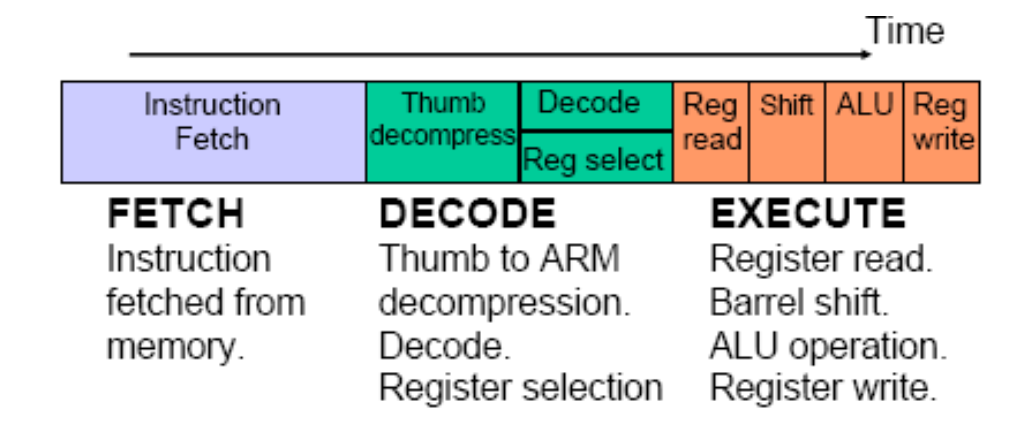
Generally the most important interrupt is assigned to the FIQ and all other interrupts are assigned to IRQ.

**Instruction Pipelines**

For the same basic speed of transistor operation, an n stage instruction pipeline allows the microprocessor to execute up to n times as many instructions in a given time.



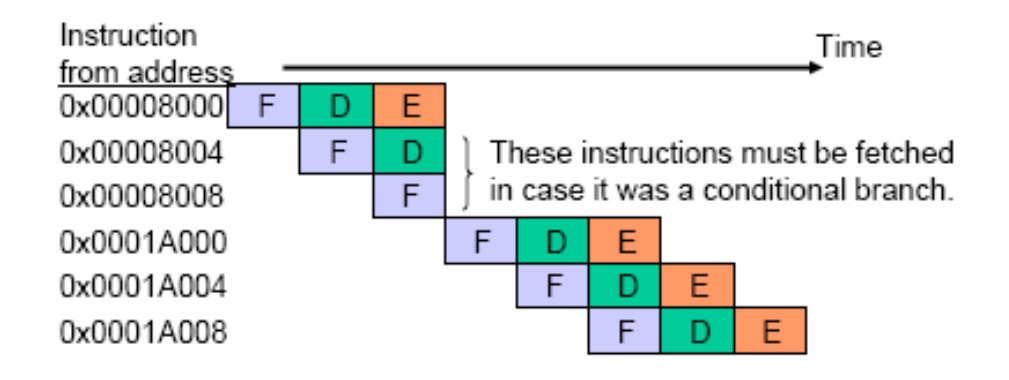
在three-stage pipline中，CPU可以同时处理执行指令，解析指令和取指令。



pipline在正常情况下，每一条指令花费1CPI/ cycle?，这是效率最高的情况。

**Branch Instruction**

branch会reload PC，所以有两个cycles被浪费了

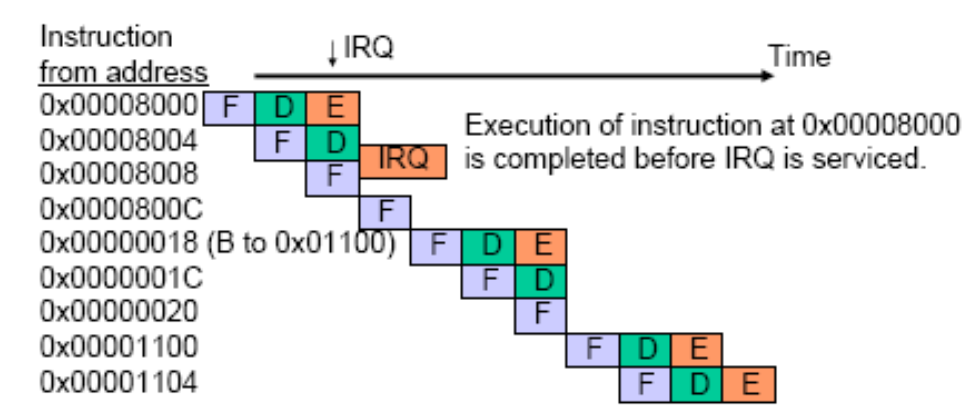


So in six clock cycles only four instructions are executed – 1.5 CPI (count from the execute stage of the first instruction to the execute stage of the last instruction)

如果条件跳转没有被执行，就不会Lost clock cost

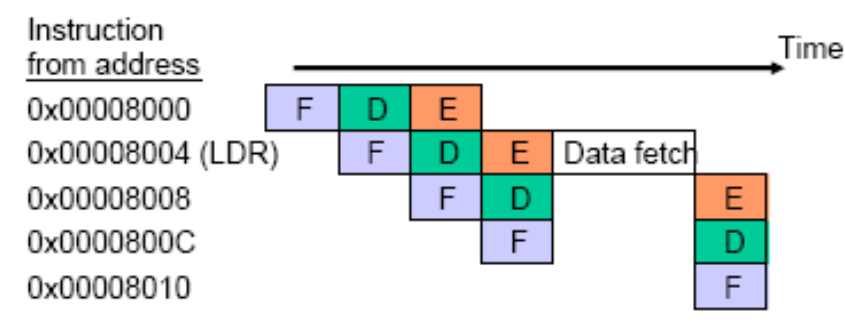
**Interrupts**

An IRQ **reloads the program counter** with 0x00000018 and then **branches** so that the pipeline is broken twice



**Load and store instruction**

用来取/存数据和取指令的data bus是同一条，所以在需要load/store data时，pipline就不能执行fetch instruction的操作。



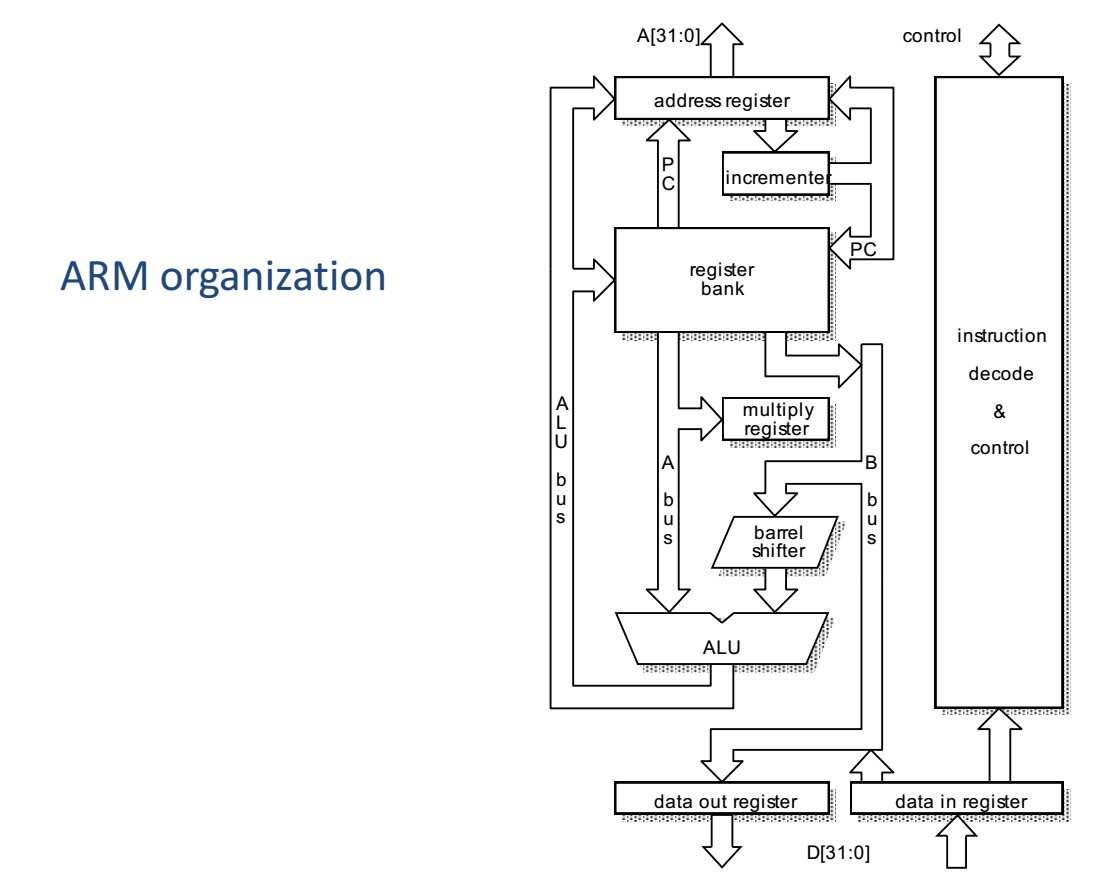
Harvard architecture使用了两条data buses来解决这个问题

von Neumann architecture 只用了一条，没有解决。

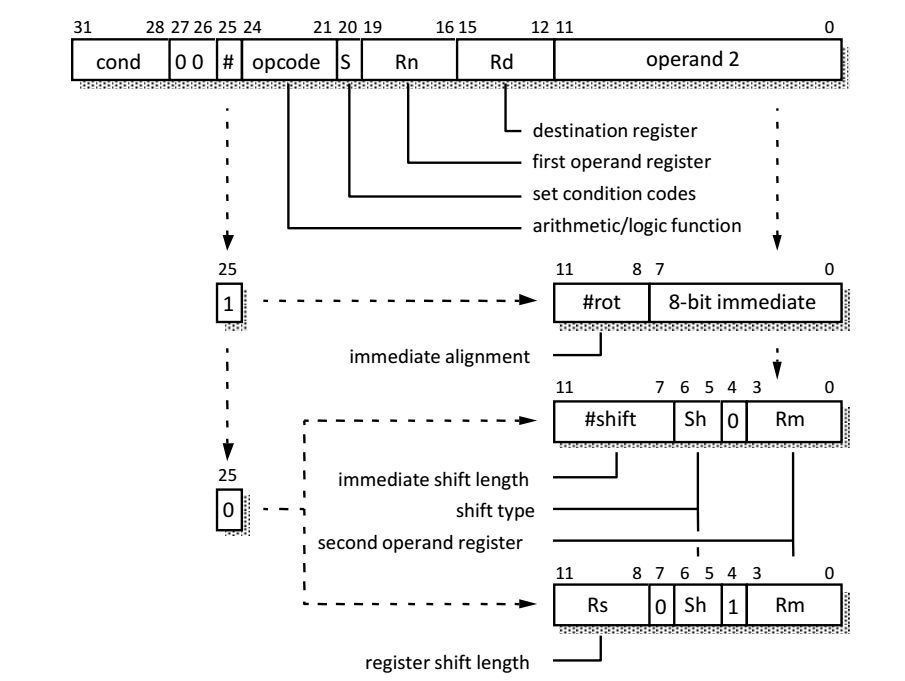
Harvard对比冯诺依曼结构来说，在data bus冲突时clock cycle lost要少，但更加复杂。

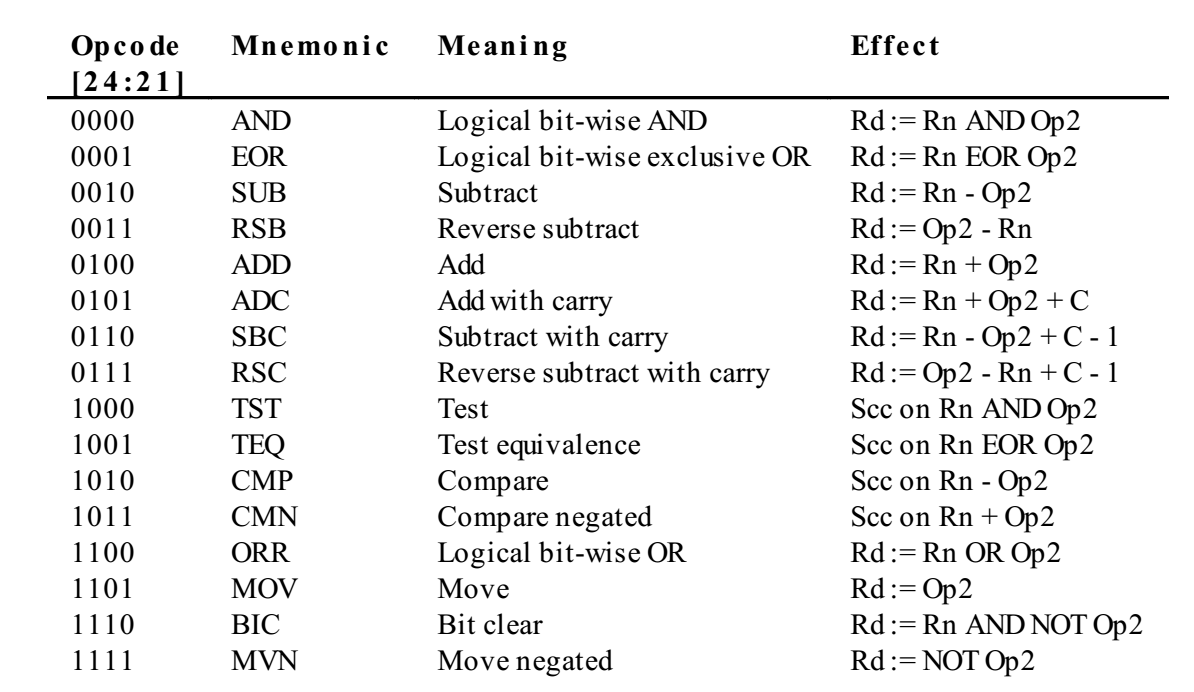
ARM9对比ARM7，添加了Harvard结构，并且是5-stage pipline

ARM7的平均CPI是1.9，ARM9是1.7

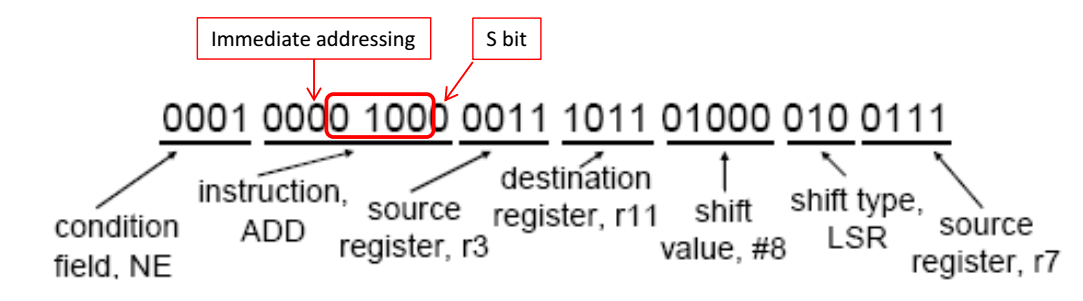


**ARM7 pipline Example**



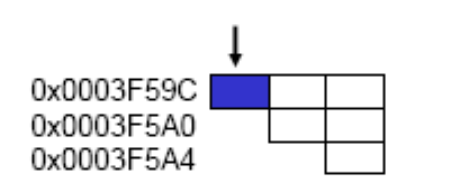


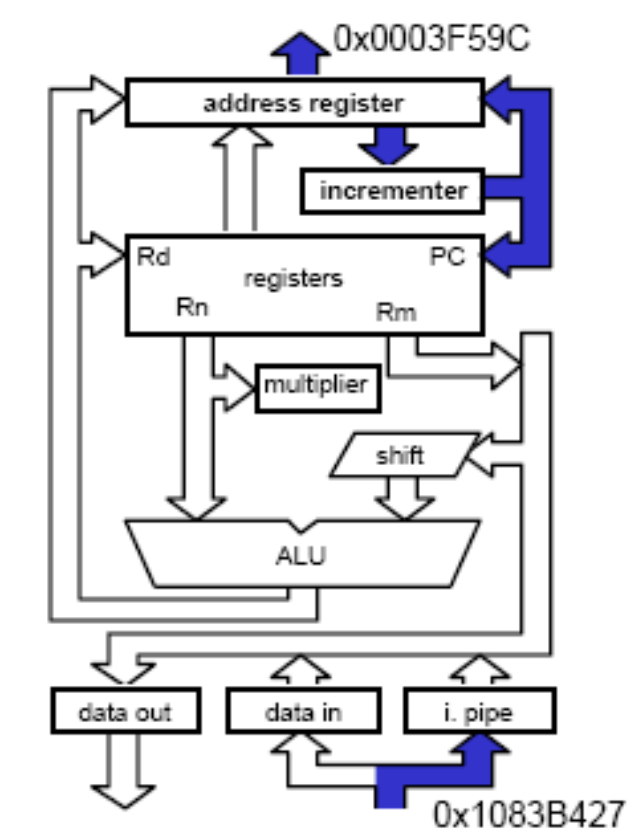
ADDNE r11, r3, r7 LSR #8 (0x1083B427)



**Fetch：**

指令0x1083B427被从address register的给的地址中取了出来，然后address register被加4，然后COPY到PC（r15）中。





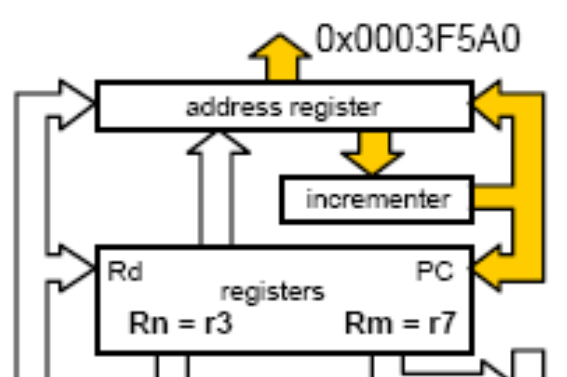
**DECODE**

首先，将16bit的Thumb Code解压缩到32bit的ARM code。

在part2中，将0x1083B427解析，condition field被检查，看这条指令是否要execute。

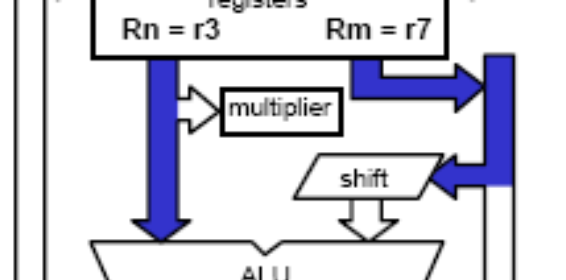
And the two source registers,

Rn = r3 and Rm = r7, are identified.(source register被识别)

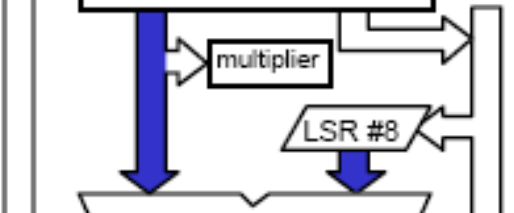


**Execute**

将source register中的值取出来，放进不同的internel data bus中



执行barrel shift



ALU中进行计算，并将结果放入ALU BUS



ALU BUS中的结果放入destination register

