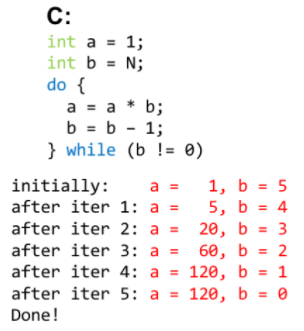
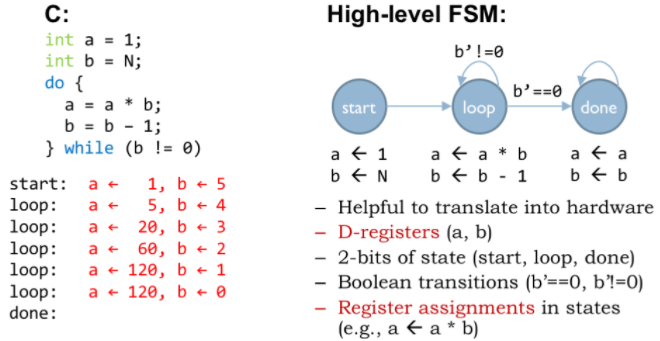
# 课件

## 例子：阶乘



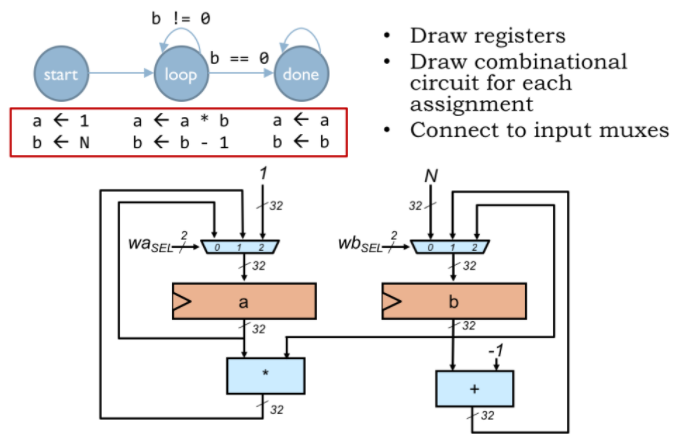
C语言实现阶乘

## 例子：阶乘2



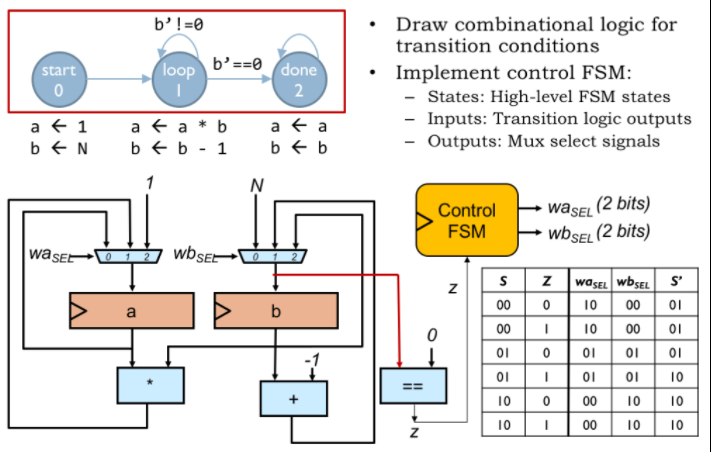
高级状态机表示C实现的阶乘

## 阶乘的数据路径

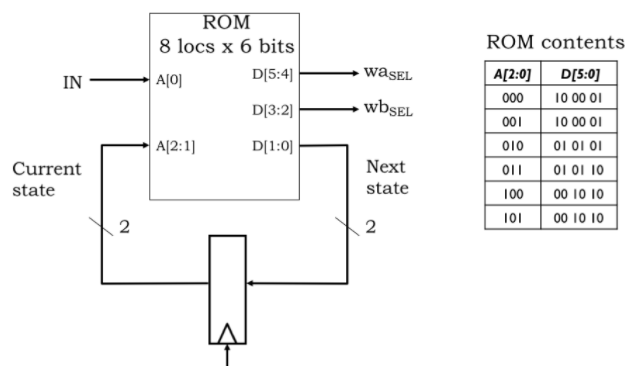


时序电路实现阶乘

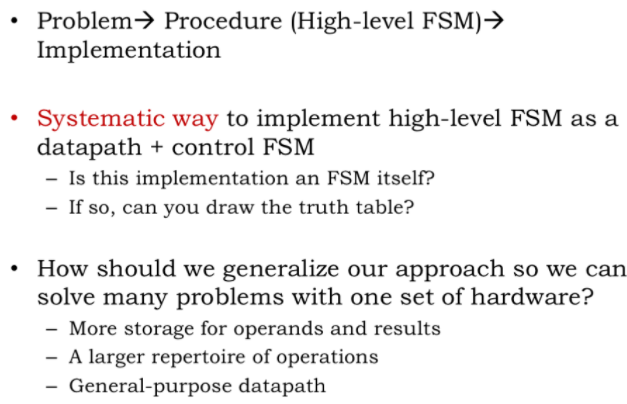
## 阶乘的控制状态机



## 控制状态机硬件



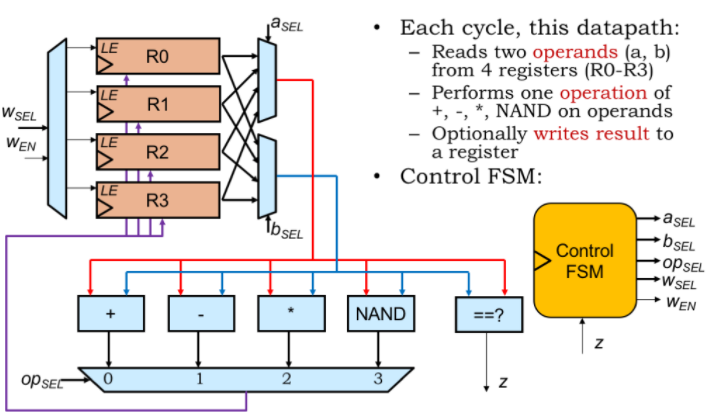
## 单目的硬件



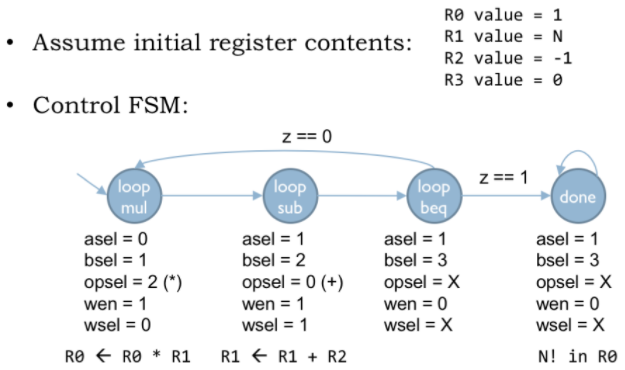
根据问题，画状态机，用电路实现它。

能否泛化我们的方案，用一组硬件解决一些问题

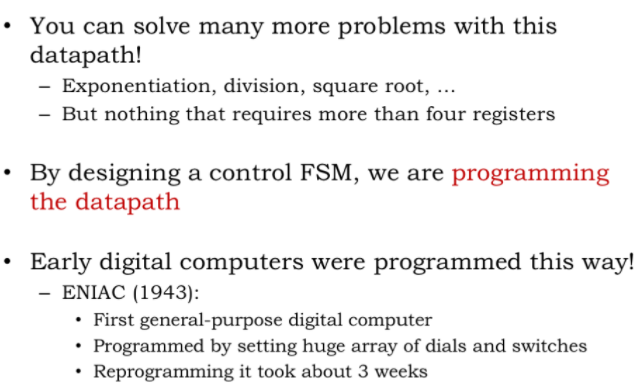
## 简单的可编程数据通路



## 阶乘的控制状态机

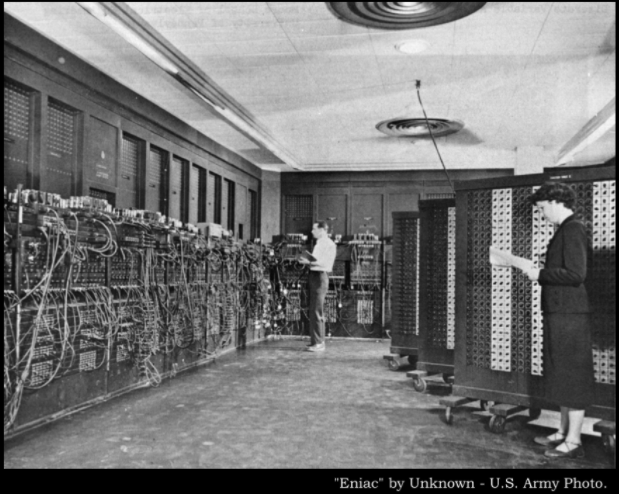


## 新问题->新控制状态机

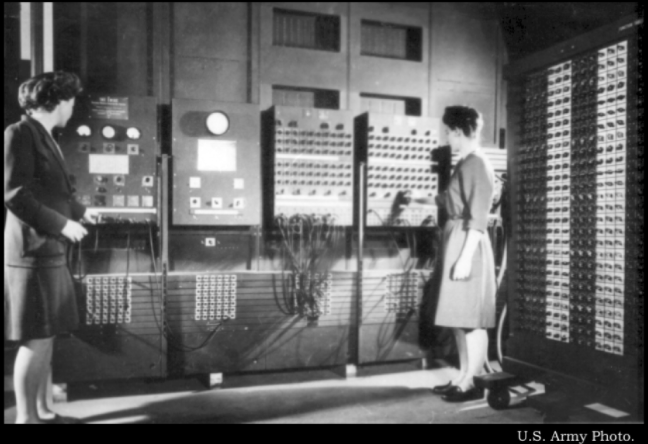


设计状态机的过程，就是在编程

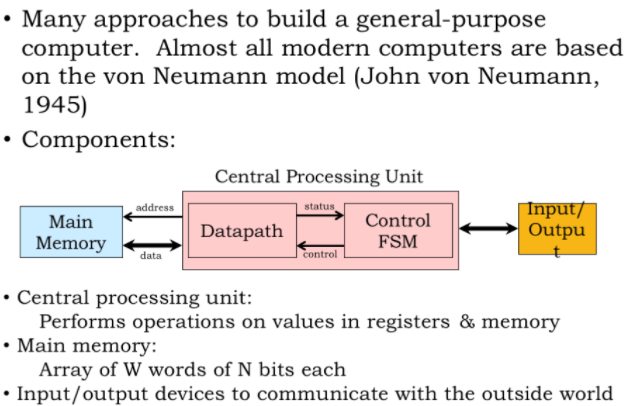
## ENIAC电脑



## ENIAC编程

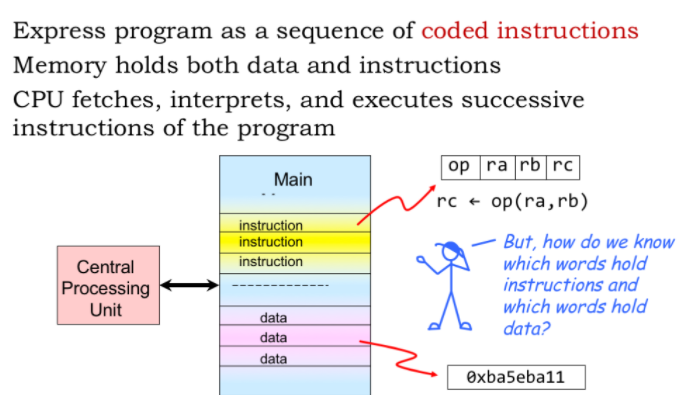


## 冯诺依曼模型



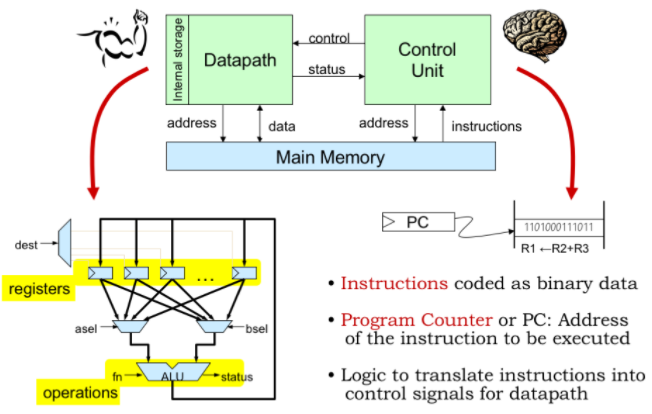
现代计算机大多基于冯诺依曼模型（CPU、主存、I/O）

## 重要想法：存储程序的电脑



主存持有数据和指令、CPU将程序解释为一系列指令、CPU获取|解释|执行程序中的连续指令

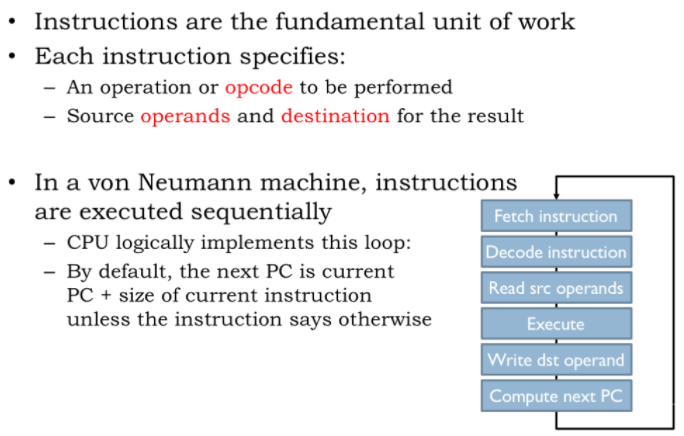
## 冯诺依曼电脑结构



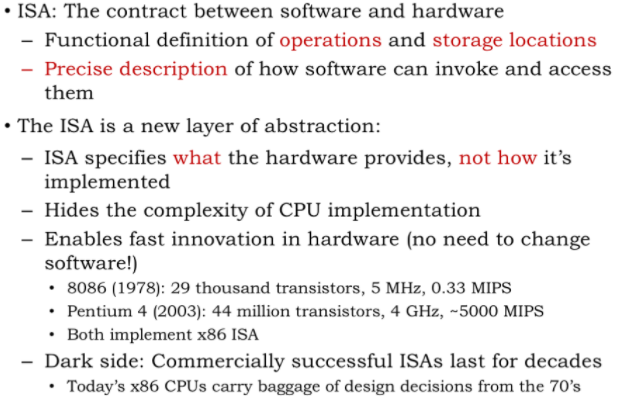
datapath中包括一些寄存器和运算操作，相当于劳工

controlUnit从主存读取指令，解释为信号传递给datapath，控制其执行，相当于大脑

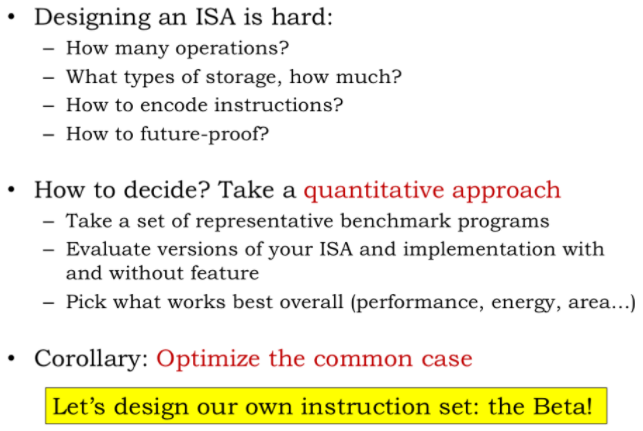
## 指令



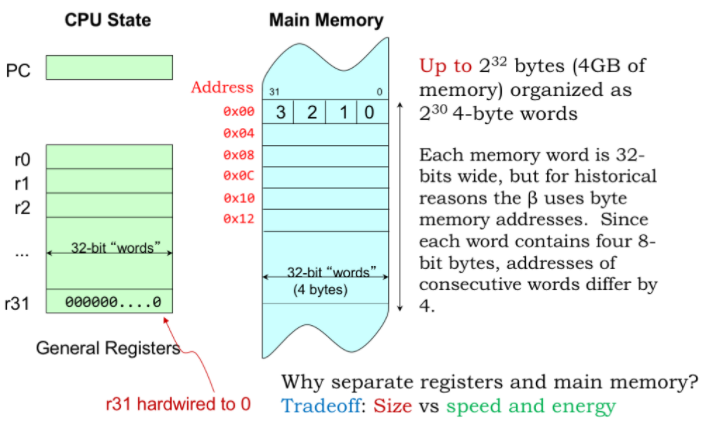
## 指令集架构



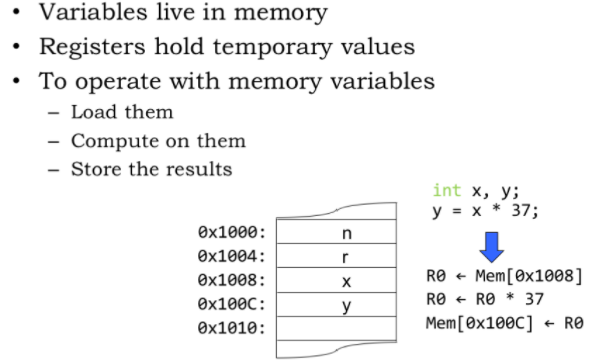
## 指令集架构设计



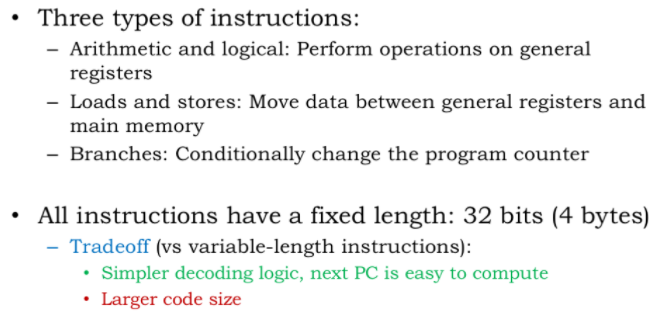
## Beta指令集：存储



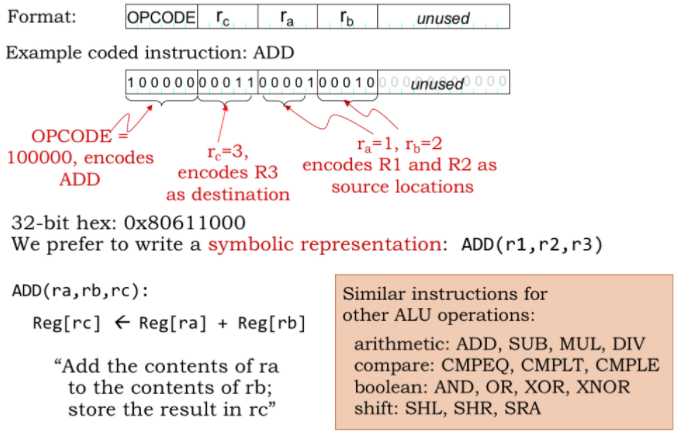
## 存储惯例



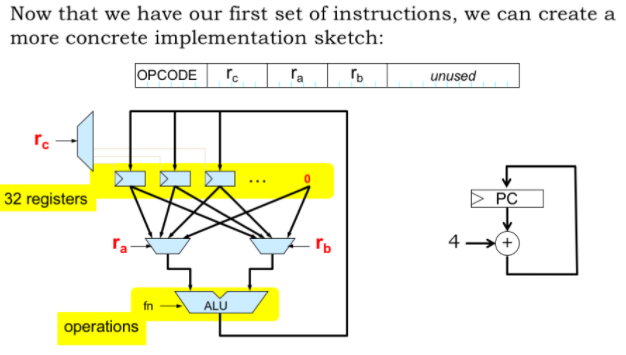
## Beta指令集：指令



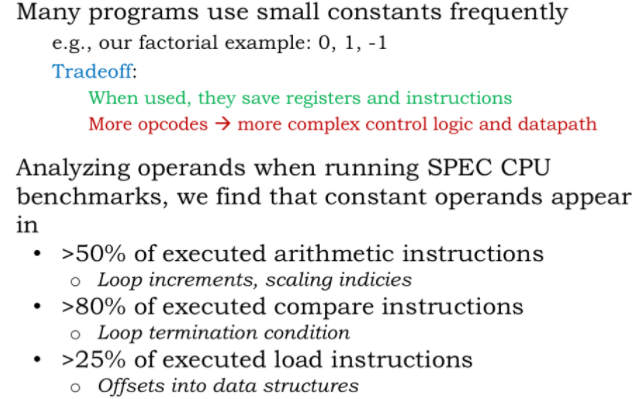
## BetaALU指令



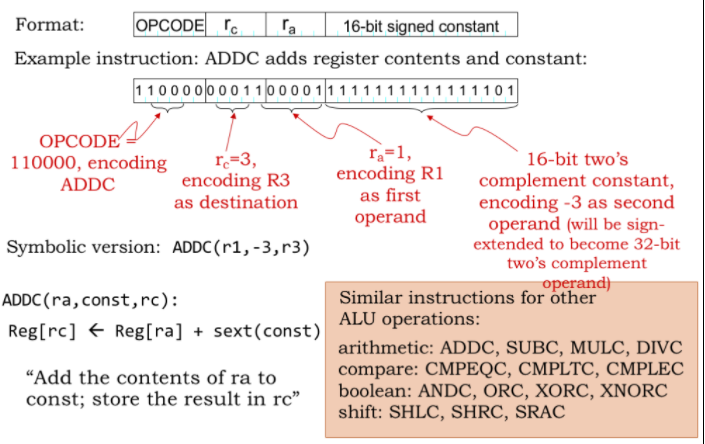
## 实现草图1



## 我们应该支持常量操作数么？



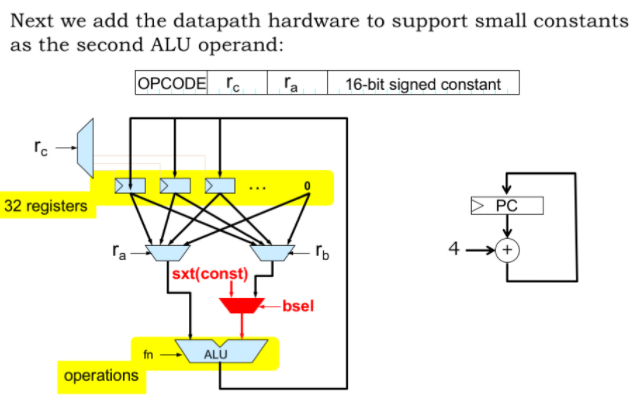
## 有常量的BetaALU指令



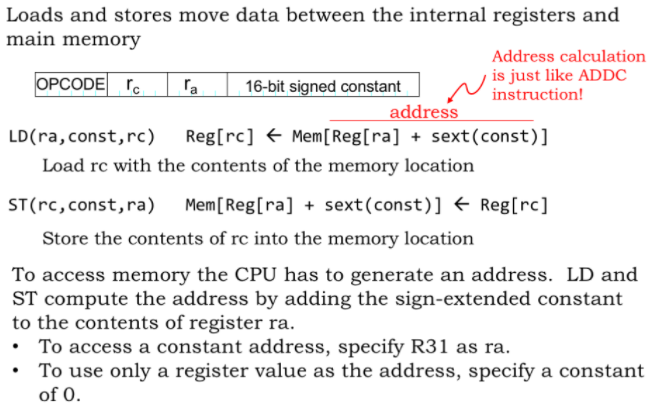
16位操作数，放到32位寄存器时，扩展符号位

如果常量超出16位能表示的数值范围，存到主存

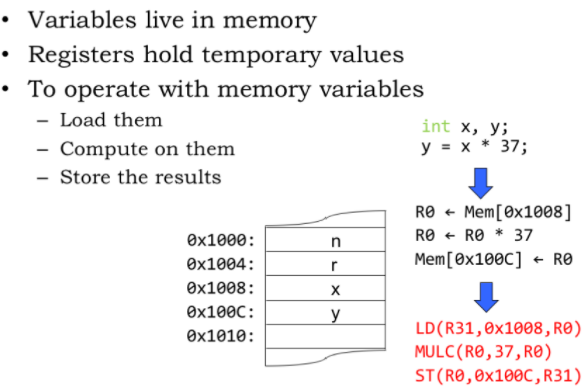
## 实现草图2



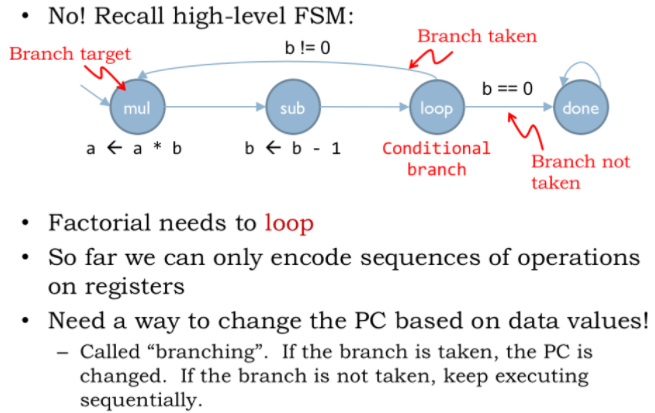
## Beta加载存储指令



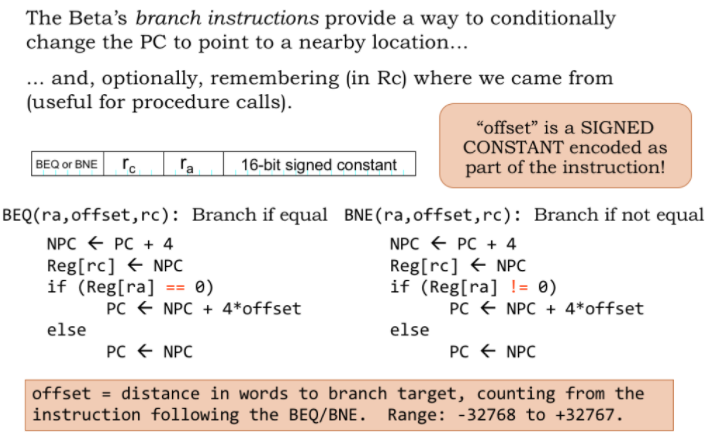
## 使用LD和ST



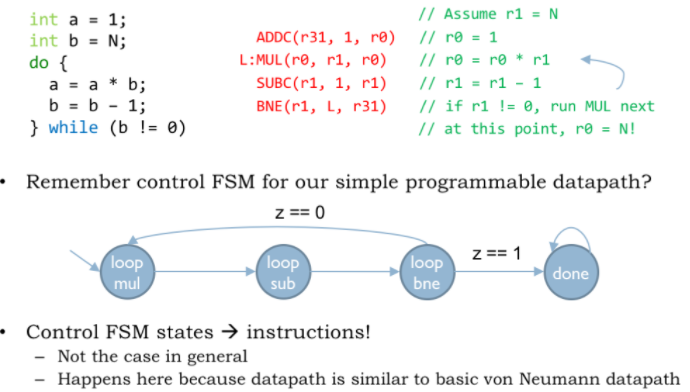
## 我们可以用ALU指令解决阶乘么



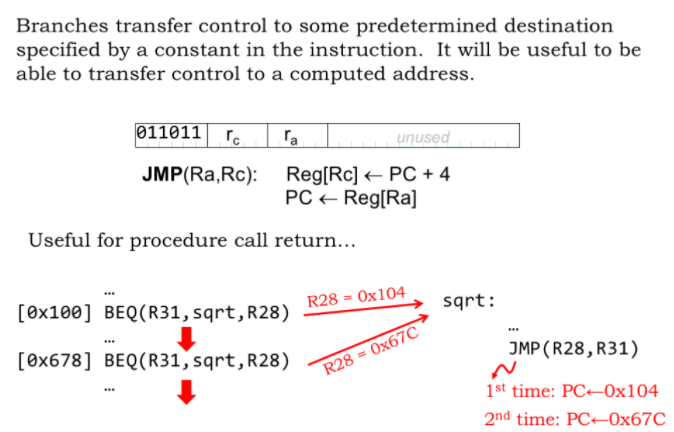
## Beta分支指令



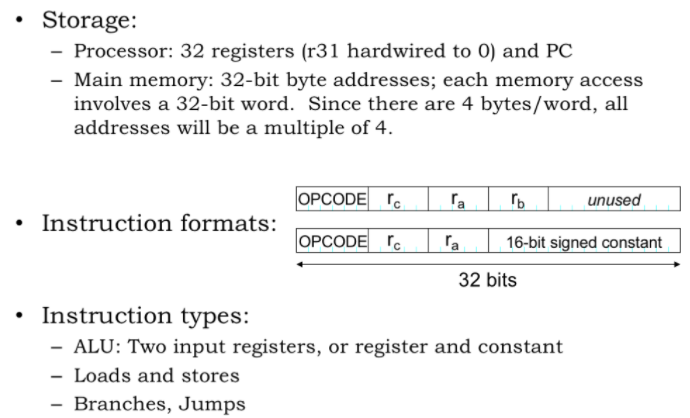
## 现在我们可以解决阶乘问题了吗？



## Beta跳转（Jmp）指令



## Beta指令集总结



# 问题

## 问题1

0x100，0x0008，ADDC(R31, 8, R2)

0x104，0x0000，ADDC(R31, 0, R1)

0x108，0x000F，ANDC(R0, 15, R3)

0x10C，0x0004，SHLC(R1, 4, R1)

0x110，00001，OR(R3, R1, R1)

0x114，0x0004，SHRC(R0, 4, R0)

0x118，0x0001，SUBC(R2, 1, R2)

0x11C，0x0002，BEQ(R2, 2, R31)

0x120，0xFFF9，BEQ(R31, -7, R31)

0x124，00001，OR(R3, R1, R1)

0x128，0x0124，LD(R31, 292, R2)

0x12C，10000，CMPEQ(R1, R2 , R1)

MEM(R31+292)存着密码，注意对输入passnumber进行了反转

## 问题2

A：SHLC(R0, 3, R1)

B：

1：0x34000000

2：22

3：20

4：10、0x1238

C：no、no

D：0x2400

E：no

## 问题3

A：0xC、0x1

B：0x83063520、4、1、0x20

C：0x35、0x1

D：0、3、0x14、0x100

E：0x87654321、0xC、0xF8765432、0x14

F：0xC、0XC0FFEE

G：0xC462003C、0xC426、0x8

H：0、0x14、0x14