北京郵電大學
BEIJING UNIVERSITY OF POSTS AND TELECOMMUNICATIONS

Computer Organization and Architecture

Chapter 20&21

Control Unit Operation & Microprogrammed Control

School of Computer Science（National Pilot Software Engineering School）
AO XIONG (熊翱)
xiongao@bupt.edu.cn

Preface

We have learned:

- Overview：
  - Basic Concepts and Computer Evolution 基本概念和计算机发展历史
  - Performance Issues 性能问题
- The computer system
  - Top level view of computer function and interconnection 计算机功能和互联结构顶层视图
  - Cache Memory cache存储器
  - Internal Memory 内部存储器
  - External Memory 外部存储器
  - Input& Output 输入输出
  - Operating System Support 操作系统支持

Preface

We have learned:

- The central processing unit：中央处理器
  - Computer arithmetic 计算机算术
  - Instruction sets 指令集
  - Processor Structure and Function 处理器结构和功能
  - Reduced Instruction Set Computers 精简指令集计算机
  - Instruction Level Parallelism and Superscalar Processors 指令级并行和超标量处理器
    - Overview of Superscalar 超标量综述
    - Design Issues of Superscalar 超标量的设计
    - Superscalar of Pentium and ARM Pentium和ARM中的超标量

Preface

We will focus the following contents today:

- **Control Unit Operation**
  - How are instructions executed in an instruction cycle? 指令如何在指令周期中执行
  - What is micro operation? In each instruction's sub cycle, how to realize its function through micro operation? 什么是微操作？在每个指令的子周期中，如何通过微操作实现其功能
  - How does the control signal of the controller control the execution of the micro operation? 控制器的控制信号如何控制微操作的执行
  - How to implement the hardwired of controller? 如何实现控制器的硬布线

Preface

- **Microprogrammed Control**
  - Why microprogram control? 为什么需要微程序控制
  - What is microprogrammed control? 什么是微程序控制
  - What is the working mechanism of microprogrammed control? 微程序控制的工作机制是什么?
  - How to sequence microprogram? 如何进行微程序定序?
  - How to execute microprogram 如何执行微程序?

Outline

- Control Unit Operation 控制器操作
  - Micro-Operations 微操作
  - Control of the Processor 处理器控制
  - Hardwired Implementation 硬布线实现方式
- Microprogrammed Control 微程序控制
  - Basic Concepts 基本概念
  - Microinstruction Sequencing 微指令定序
  - Microinstruction Execution 微指令执行

The function of a processor 处理器的功能

- Instruction Fetch and Execute 取指和执行指令
  - Execute program 执行程序
- Interrupts 中断
  - Handling performance differences between CPU and other components 处理CPU和其他部件的性能差异
- I/O Function I/O功能
  - Interworking with peripherals 和外设互通

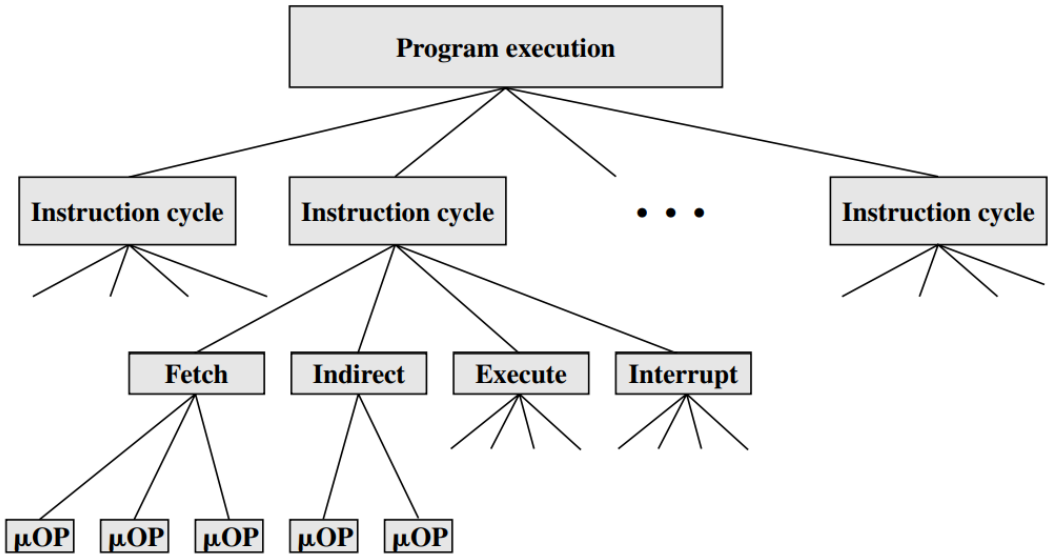
Composition of instructions 指令的组成

- Instructions include opcodes and operands 指令包括操作码和操作数
- Opcodes 操作码
  - Determines what type of operation the instruction does 确定指令做什么类型的操作
- Operands 操作数
  - Determine the object of instruction operation 确定指令操作的对象
  - Operands may be in registers, memory, I/O, or immediate 操作数据可能在寄存器、内存、I/O，或者立即数
  - Finding operands through addressing mode 通过寻址模式找到操作数

Micro-operations 微操作

- A computer executes a program to complete user specified functions 计算机执行程序来完成用户指定的功能
  - Program contains many instructions 程序包括若干指令
  - Instruction execution includes several cycles, such as fetch cycle, execution cycle, indirect cycle, etc 指令执行包括若干个周期，如取指周期、执行周期、间接周期等
- Each cycle has a number of steps 每个周期有一系列步骤
  - Called micro-operations 称为微操作
  - Each step does very little 每一步做很小的操作
  - Atomic operation of CPU CPU的原子操作

Constituent of program 程序组成

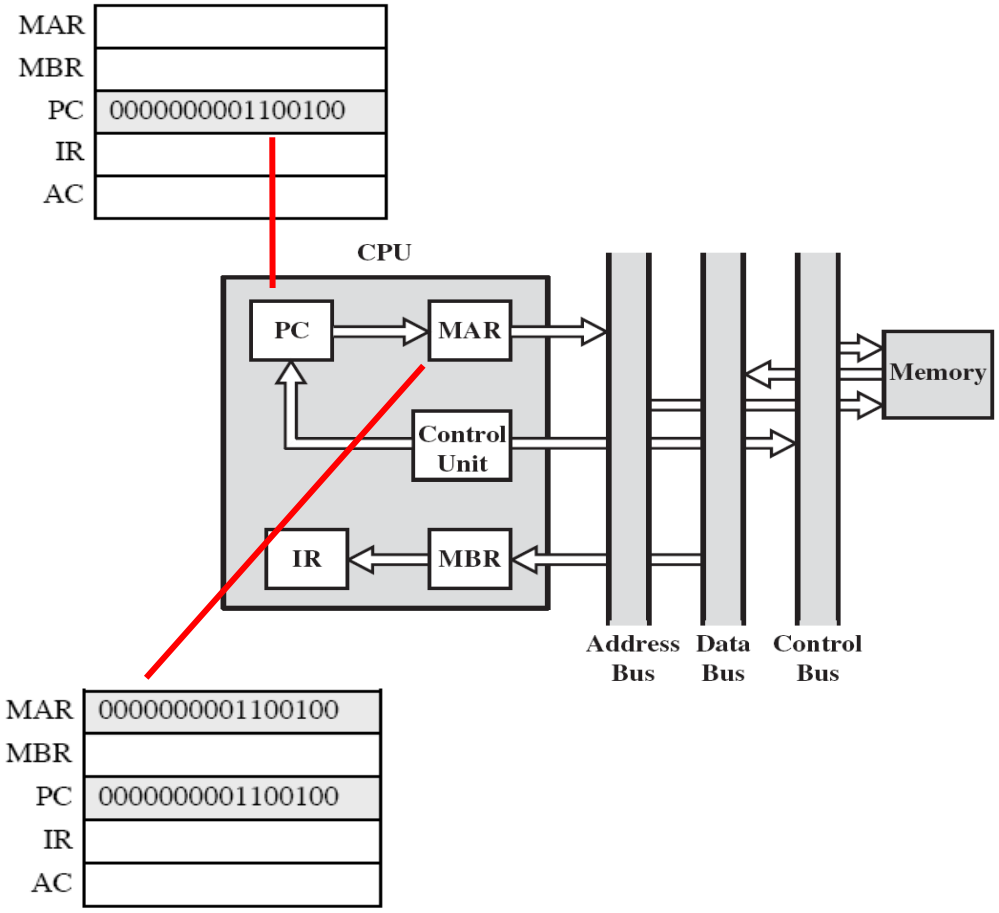


- 程序的执行是由若干个指令执行组成的
- 每个指令的执行就是一个指令周期
- 指令周期由若干个子周期组成，比如取指周期、执行周期等
- 每个子周期包含若干个更小的操作，这些操作称为微操作

Fetch 取指

- Program Counter (PC) 程序计数器
  - Holds address of next instruction to be fetched 下一指令地址
- Memory Address Register (MAR) 存储器地址寄存器
  - Connected to address bus 连到地址总线
  - Specifies address for read or write op 指定读写操作数的地址
- Memory Buffer Register (MBR) 存储器缓冲寄存器
  - Connected to data bus 连到数据总线
  - Holds data to write or last data read 保存读写数据
- Instruction Register (IR) 指令寄存器
  - Holds last instruction fetched 保存最新取到的指令

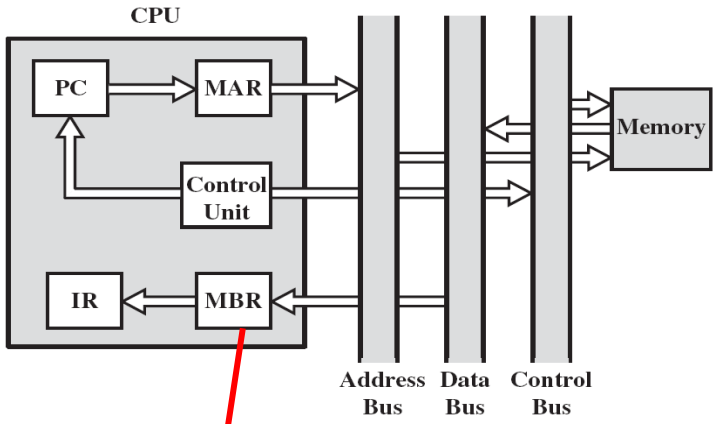
Step1



t1: MAR ← (PC)

- 下一个指令的地址是放在PC里
- MAR是与地址总线连接的唯一寄存器
- 取指的第一步是PC把下一个指令的地址给MAR
- 经过第一步之后，PC寄存器的内容复制到MAR寄存器中

Step 2



t2

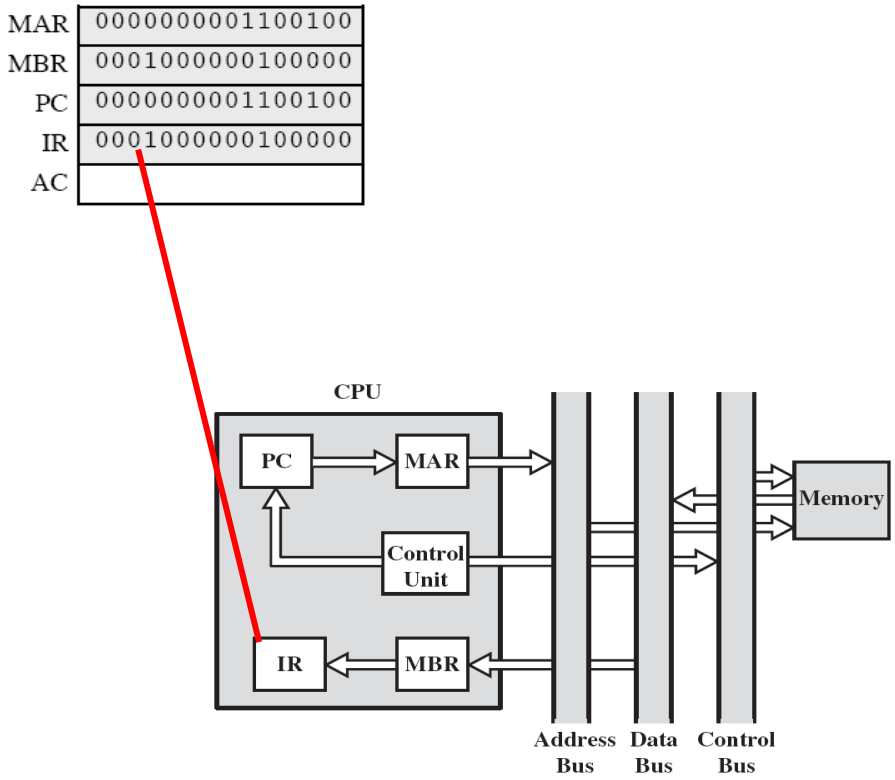
MAR	0000000001100100
MBR	0001000000100000
PC	0000000001100100
IR	
AC	

t2: MBR ←Memory
PC ← (PC) + I

- MAR收到地址后，把地址放到地址总线上
- 控制单元发一个读命令到控制总线上
- 存储器收到读命令后，根据地址读出指令内容，并放到数据总线
- 数据总线上的数据读到MBR
- PC中还需要递增一个指令长度，以得到下一个指令的地址

Step 3

t3



- MBR的内容传送给IR
- 传送完成后，MBR就可以释放，用于下一步操作
- IR中保存的就是下一个需要执行的指令
- 通过这几个微操作，完成了取指

t3: IR ← (MBR)

Fetch sequence 取指序列

- t1: MAR <- (PC)
- t2: MBR <- (memory) PC <- (PC) +I
- t3: IR <- (MBR)

or

- t1: MAR <- (PC)
- t2: MBR <- (memory)
- t3: PC <- (PC) +1 IR <- (MBR)

t1~t3：时钟周期

- 取指周期实际上由三步、四个微操作组成。每个微操作都涉及到数据在寄存器之间的流动。
- 如果数据的流动是独立的，这些操作可以在同一个时钟周期内完成
- PC递增微操作，既可以在t2完成，也可以在t3完成

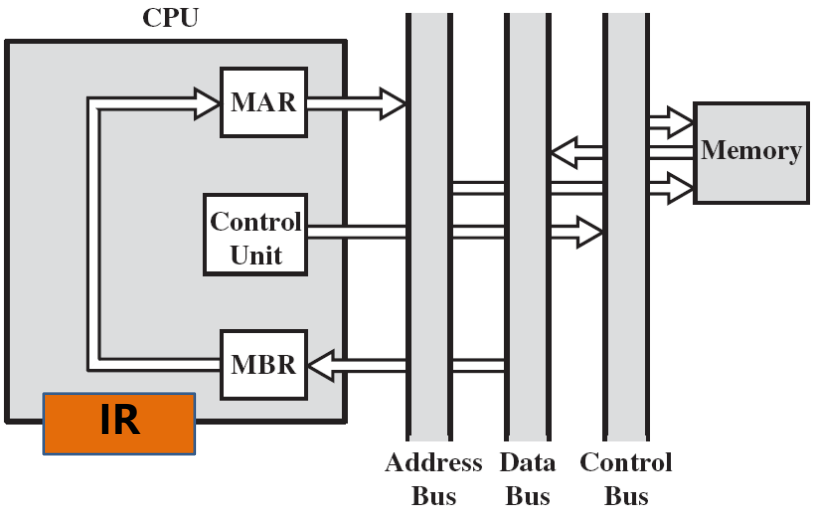
Rules for Micro-operations grouping 微操作分组原则

- Proper sequence must be followed 事件的顺序必须恰当
  - MAR <- (PC) must precede MBR <- (memory) MAR <- (PC) 必须先于MBR <- (memory)
- Conflicts must be avoided 必须要避免冲突
  - Must not read & write same register at same time 不能在同一时刻读写同一个寄存器
  - MBR <- (memory) & IR <- (MBR) must not be in same cycle MBR <- (memory) 和 IR <- (MBR) 不能在同一个时间单位
- Also: PC <- (PC) +1 involves addition PC递增涉及到加法
  - Use ALU 用ALU
  - May need additional micro-operations 可能会需要额外的微操作

Indirect cycle 间接周期

- In indirect addressing, it is necessary to get the address of the operand from the memory first. This process is called indirect cycle 间接寻址时，需要先去存储器中得到操作数的地址，这个过程称为间接周期
- Instruction contains the address of storage location where the operand address is located 指令中包含了操作数地址所在的存储单元的地址
- The goal of an indirect cycle is to get the address of the operand into the MBR 间接周期的目标是将操作数的地址取到IR中
  - MBR contains operand address and put into IR MBR中包含操作数地址，并传给IR
  - IR is now in same state as if direct addressing had been used 这个时候IR的状态与直接寻址是一样的

Indirect cycle diagram 间接周期图示



- 间接周期包括3个微操作，需要三个时间单位来完成
- T1：IR将地址域内容给MAR
- T2：存储器将地址信息给MBR
- T3：MBR把操作数的地址放到IR的地址域
- 这三个时间单位的顺序不能乱，也不能并行，因为都存在操作的顺序关系

t1: MAR ← (IR(address))

t2: MBR ← Memory

t3: IR(address) ← (MBR(address))

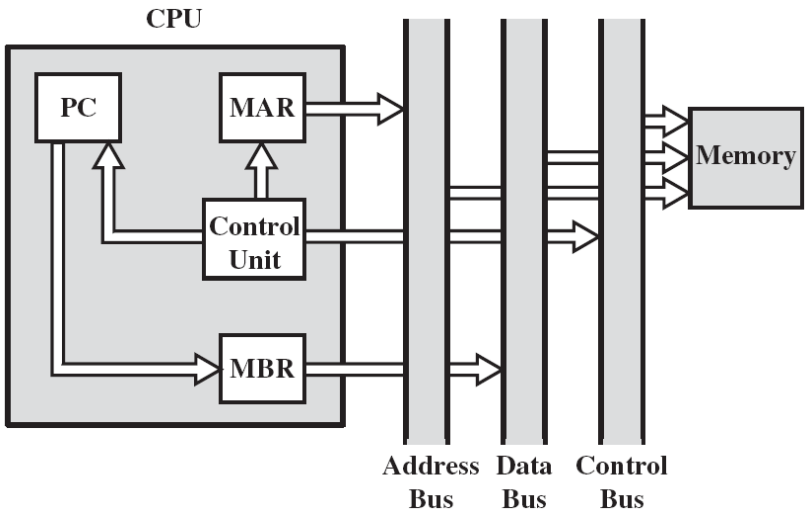
Interrupt cycle 中断周期

- After instruction processing is completed, check whether there is an interrupt 指令处理完成后，需要检查是否有中断产生
- If an interrupt occurs, enter the interrupt cycle 如果有中断发生，进入中断周期
- The interrupt cycle is actually to stop executing the original instruction stream and execute the interrupt handler first. 中断周期实际上是停止执行原定的指令流，先去执行中断处理程序
- After the interrupt handler completes processing, continue to process the original instruction 中断处理程序处理完成后，继续处理原来的指令
  - Question：What needs to be done? 问题：需要做哪些操作呢？？

Interrupt cycle 中断周期

- Interrupt cycle needs to complete these operations 中断周期需要完成这些操作
  - The instruction address before the interrupt needs to be saved 保存中断前的指令地址
  - Get the instruction address of the start of the interrupt handler to the PC 获取中断处理程序起始的指令地址给PC
- After the PC gets the interrupt processing address, it enters the index retrieval cycle PC得到中断处理地址后，进入取指周期

Interrupt cycle diagram 中断周期图示



- 中断周期有4个微操作，需要三个时间单位
  - T1：PC将下一个指令地址给MBR
  - T2：操作系统将保存PC的存储器地址信息给MAR。同时，中断处理程序把起始地址给PC
  - T3：MBR把PC之前给它的地址存入存储器
- 对于多种或多级中断，可能会需要额外的微操作来取得保存地址和子程序的起始地址
- 程序执行的上下文的保存，属于中断处理程序需要完成的功能，而不是微操作需要完成的功能

t1: MBR ← (PC)

t2: MAR ← save_address

PC ← Routine_address

t3: Memory ← MBR

Execute cycle 执行周期

- Fetching, indirect and interruption cycle can be determined in advance 取指、间接、中断周期可以预先确定
  - Each cycle contains a fixed sequence of micro operations 每个周期都包含了固定序列的微操作
- For execution cycle 对于执行周期
  - Different operation codes correspond to different operations 不同的操作码对应不同的操作
  - Each operation has a specific micro operation sequence 每个操作都有特定的微操作序列

Example–ADD

- ADD R1,X
  - add the contents of location X to Register 1 , result in R1 将X位置的内容和R1相加，存入R1
- Micro-operation sequence is as follows 微操作序列
  - t1: MAR <- ($IR_{address}$) IR把操作数的地址给MAR
  - t2: MBR <- (memory) 控制单元发出读命令，存储器把操作数给MBR
  - t3: R1 <- R1 + (MBR) ALU把R1和MBR的值相加，然后存到R1中
- The above micro-operation execution cannot overlap 上述微操作执行不能重叠

Example–ISZ

- ISZ X
  - increment and skip if zero　X增加1，如果为0就跳步
- Micro-operation sequence is as follows　微操作序列
  - t1: MAR <- ($IR_{address}$)　IR中的地址部分给MAR
  - t2: MBR <- (memory)　存储器把X地址中的数据给MBR
  - t3: MBR <- (MBR) + 1　MBR中的数据+1
  - t4: memory <- (MBR)　把MBR的数据存到存储器中

    if (MBR) == 0 then PC <- (PC) + 1　MBR如果为0，PC递增1
- Notes:
  - if is a single micro-operation　if是一个微操作，判断和PC递增作为1个微操作来完成
  - Micro-operations can done during t4 in parallel　t4的微操作可以并行完成

Example–BSA

- BSA X
  - Branch and save address 转移并保存地址
  - Address of instruction following BSA is saved in X BSA之后的指令地址保存在X中
  - Execution continues from X+1 由X+1地址开始执行
- Micro-operation sequence is as follows 微操作序列
  - t1: MAR <- ($IR_{address}$) IR的地址部分给MAR
    MBR <- (PC) PC的内容给MBR
  - t2: PC <- ($IR_{address}$) IR的地址部分给PC
    memory <- (MBR) MBR的内容放到存储器单元中
  - t3: PC <- (PC) + 1 PC递增1个指令单元
- T1、T2完成将下一个指令的地址保存的功能。T2、T3，将X+1的地址给了PC。下一个要执行的指令就是X+1

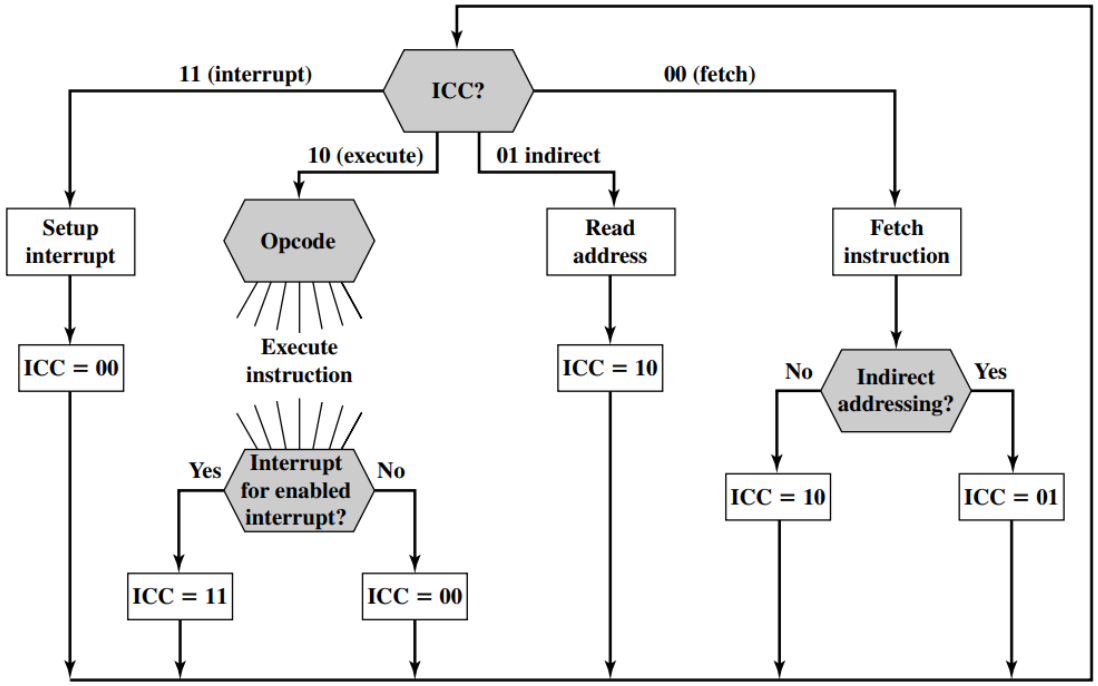
Instruction cycle -1 指令周期1

- Each phase decomposed into sequence of elementary micro-operations 每个阶段分解为基本微操作序列
  - E.g. fetch, indirect, and interrupt cycles 例如，取指、间接和中断周期
- Execute cycle 执行周期
  - One sequence of micro-operations for each opcode 每个操作码都有一个微操作序列

Instruction cycle -2 指令周期2

- Need to tie sequences together 需要将微操作序列连起来
- Assume new 2-bit register 假定了一个2位的寄存器
  - Instruction cycle code (ICC) designates which part of cycle processor is in 指令周期代码（ICC）指定当前处于处理器周期的哪一部分
    - 00: Fetch 取指
    - 01: Indirect 间接
    - 10: Execute 执行
    - 11: Interrupt 中断
  - After each stage, corresponding settings will be made 每个阶段结束后，会进行相应的设置

Flowchart for instruction cycle 指令周期流程图



- 取指周期后，看是否是间接寻址。如果是间接寻址，进入01间接周期，如果不是，进入执行周期。
- 间接周期后，读取地址，然后进入10执行周期
- 执行周期，根据操作码来确定做什么操作。然后判断是否有中断。如果有中断，进入11中断周期。如果没有，进入00取指周期。
- 中断周期，设置中断，然后进入中断处理程序的取指周期，继续取指

Control Unit Operation 控制器操作

- Control of the Processor 处理器控制
  - Functional Requirements 功能要求
  - Control Signals 控制信号
  - A Control Signals Example 控制信号举例
  - Internal Processor Organization 处理器内部组织

Functional requirement 功能要求

- Micro-operations are the most basic function of the processor

  微操作是处理器最基本的功能

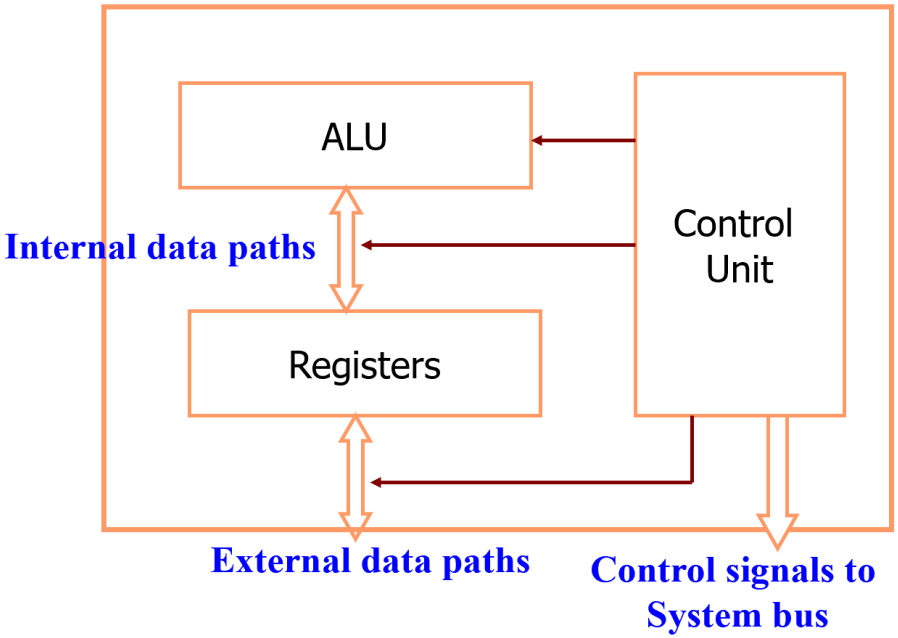
- Controller needs to control the processor to complete micro operation控制器需要对处理器进行控制，完成微操作

- Three elements of the controller include: 控制器的三要素包括:
  - Basic elements of processor 处理器的基本元素（控制对象）
  - Micro-operations that processor performs 微处理器执行的操作（做什么）
  - How to control 如何去控制（怎么做）

Basic elements of processor 处理器的基本元素

- ALU 算术逻辑单元
  - Core of CPU, perform computing function CPU的核心，完成计算功能
- Registers 寄存器
  - Temporarily save data inside CPU CPU内部暂存数据
- Internal data paths 内部数据通道
  - Data transfer inside CPU CPU 内部的数据传输
- External data paths 外部数据通道
  - Transfer between CPU and memory, I/O CPU和内存、I/O之间的传输
- Control Unit 控制单元
  - Control various operations inside the CPU 控制CPU内部的各个操作

Basic elements of processor 处理器的基本元素



- 控制单元负责CPU中各个元素的控制，包括ALU、寄存器以及内部和外部通道
- 控制单元还发控制信号给系统总线，用于CPU和存储器或I/O之间的数据传输。
- ALU从寄存器中提取数据进行处理，之间通过内部数据通道交换数据
- 寄存器通过外部数据通道，和存储器或I/O模块进行数据交换

Types of micro-operation 微操作类型

- execution of instructions consists of several stages 指令的执行包括若干个阶段
- Each stage can be subdivided into several micro bit operations 每个阶段又可以细分为若干个微位操作
- Micro-operations can be divided into the following four categories 微操作可以分为如下四类：
  - Transfer data between registers 寄存器之间的数据传送
  - Transfer data from register to external 将数据从寄存器传输到外部
  - Transfer data from external to register 将数据从外部传输到寄存器
  - Perform arithmetic or logical ops 执行算术或逻辑运算

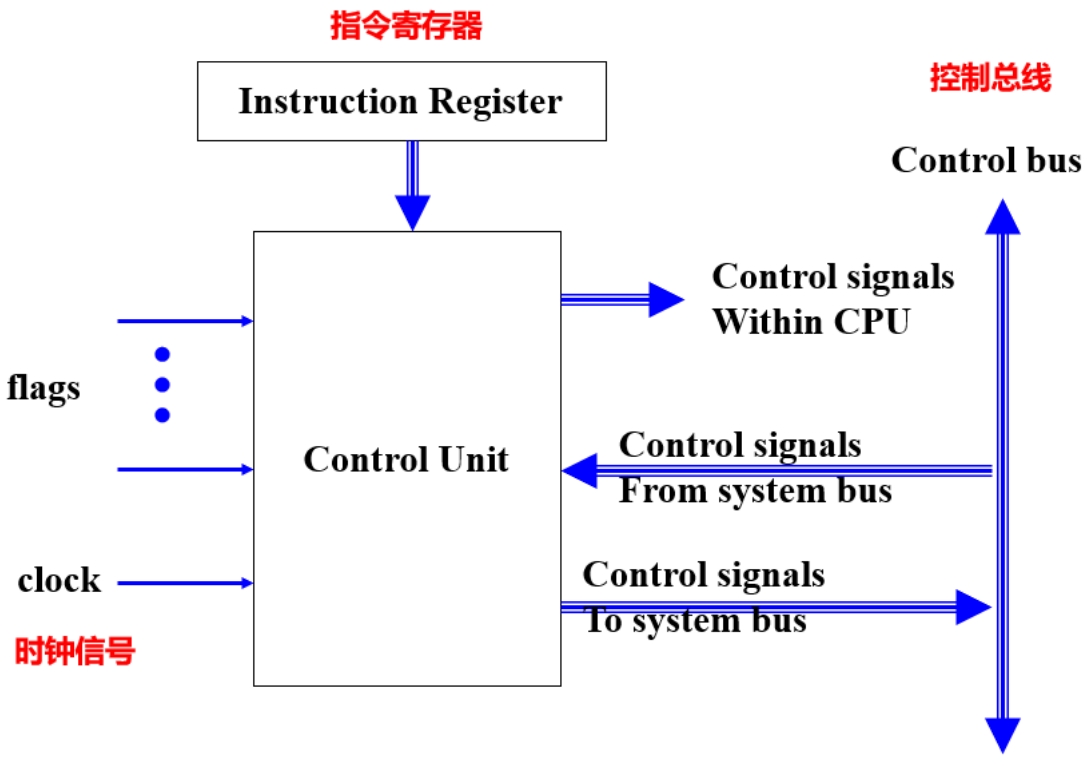
Functions of control unit 控制器的功能

- Operation object and operation type have been determined 操作对象和操作类型已经确定
- Function of the controller is to determine how to do it 控制器的功能是确定如何去做
- Sequencing 定序
  - Micro-operations are sequential 微操作有前后关系
  - Causing the CPU to step through a series of micro-operations 使CPU按照顺序执行一系列微操作
- Execution 执行
  - Causing the performance of each micro-op 控制完成每个微操作
- This is done using Control Signals 通过控制信号实现

Control signals 控制信号

- For control unit to perform its function 为了使控制单元执行功能
  - Inputs：show the state of the system 输入：表示系统的状态
  - Outputs：control the behavior of the system 输出：控制系统的行为
  - External interface，called external specifications 输入和输出，为控制器对外的接口，称为外部规范
- For control unit itself 对于控制器本身
  - it must have the logic to perform its sequencing and execution functions 它必须具有执行定序和执行功能的逻辑
  - Called internal specifications 称为内部规范

Model of control unit 控制器模型



- 控制器的输入包括：时钟信号，各种标志，指令寄存器中的指令，来自控制总线的控制信号（中断）
- 控制器的输出包括：CPU内部的控制信号，到控制总线的控制信号
- 控制器内部就是如何根据输入，来生成控制信号

Control signals – 1 控制信号1

- Clock 时钟
  - Provide clock signal for timing of controller 提供时钟信号用于定时
  - One micro-instruction (or set of parallel micro-instructions) per clock cycle 每个时钟周期一条微指令（或一组并行微指令）
- Instruction register 指令寄存器
  - Op-code for current instruction 当前指令的操作码
  - Addressing mode 寻址方式
  - Determines which micro-instructions are performed 确定执行哪些微指令

Control signals -2 控制信号2

- Flags 标志
  - State of CPU CPU的状态
  - Results of previous operations 上一个操作的结果
- Control signal from control bus 从控制总线过来的控制信号
  - Interrupts 中断信号
  - Acknowledgements 应答信号

Control signals – output 控制信号-输出

- Within CPU CPU内部
  - Cause data movement 引起数据移动
  - Activate specific functions 激活特别的功能
- Via control bus 通过控制总线
  - To memory，control read or write 到存储器，控制读写操作
  - To I/O modules 到I/O模块

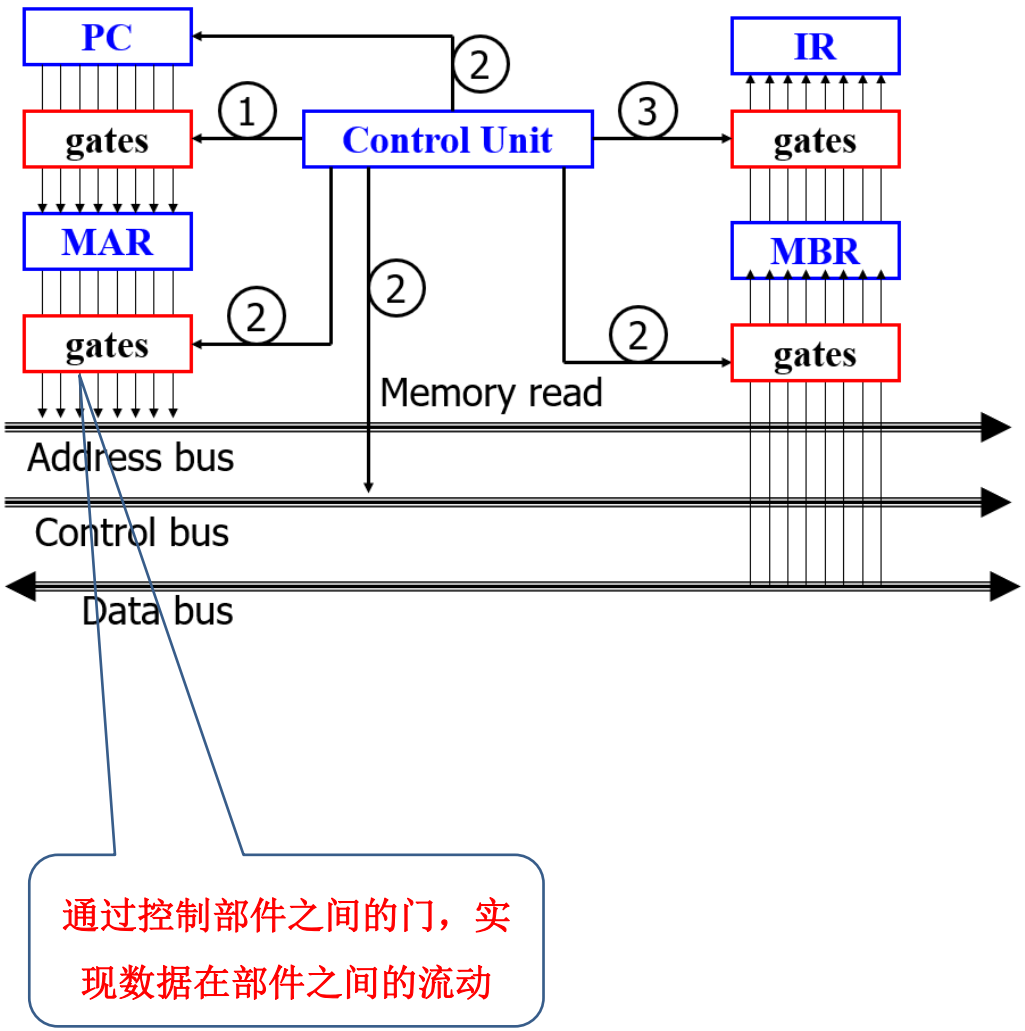
Control signals – output 控制信号-输出

- **It generates three types of output control signals 产生三种类型的输出控制信号**
  - Data paths: the signals control the internal flow of data 数据通道：控制内部数据流动的控制信号
  - ALU: the signals control the operation of the ALU ALU：控制ALU的操作的控制信号
  - System bus: the control unit sends these signals to the control lines of the system bus 系统总线：控制单元发送控制信号到系统总线中的控制线

Example control signal sequence – Fetch 取指举例

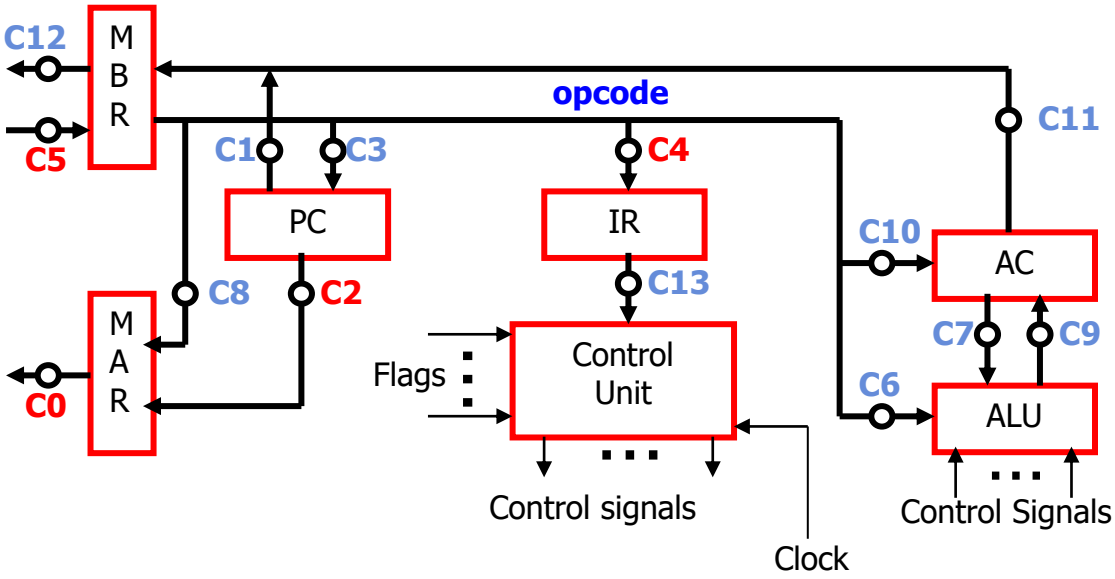
- MAR <- (PC)
  - Control unit activates signal to open gates between PC and MAR
    控制单元激活信号，打开PC和MAR之间的闸门
- MBR <- (memory)
  - Open gates between MAR and address bus 打开MAR和地址总线之间的门
  - Memory read control signal 存储器读取控制信号
  - Open gates between data bus and MBR 打开数据总线和MBR之间的门
  - PC<- (PC+1)

Example: the fetch cycle 取指周期举例



- 第一步：控制器发信号给PC和MAR之间的逻辑门，PC指令的地址就到MAR
- 第二步：控制器打开MAR和地址总线的门，将地址放到地址总线，发出存储器读信号到控制总线，打开数据总线和MBR之间的逻辑门。存储器收到读命令后，将地址对应的内容放到数据总线，MBR从数据总线得到指令内容。PC的内容加1的操作也可以在这一步完成
- 第三步：控制器打开MBR和IR之间的逻辑门，MBR中的指令就到了IR

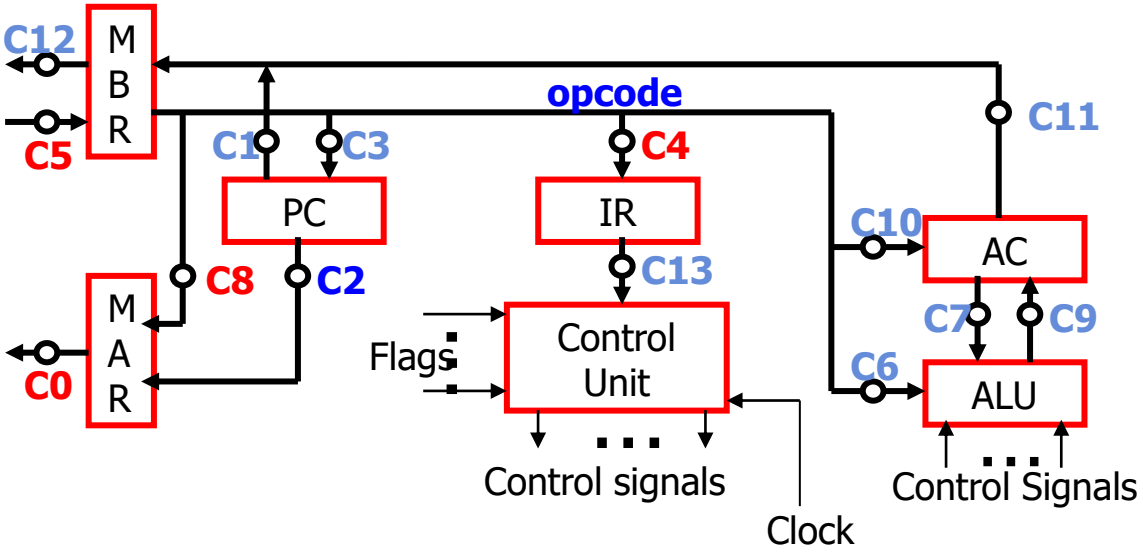
Fetch cycle 取指周期



Fetch: t1: MAR ← (PC) C2
t2: MBR ← Memory C0,C5
PC ← (PC) + I
t3: IR ← (MBR) C4

- T1周期：PC的内容给MAR，打开C2，PC的地址到了MAR。
- T2周期：读内存。打开C0，地址到地址总线；打开C5，数据总线的内容到MBR中
- T3周期：MBR中的内容给IR，需要打开C4。
- 控制信号没有在图上

Indirect cycle 间接周期



- T1周期：取指中的地址部分给MAR，打开C8
- T2周期：存储器中的数据到MBR。需要打开C0，C5，让数据总线的内容到MBR中
- T3周期：MBR中的地址给IR中的地址部分，打开C4

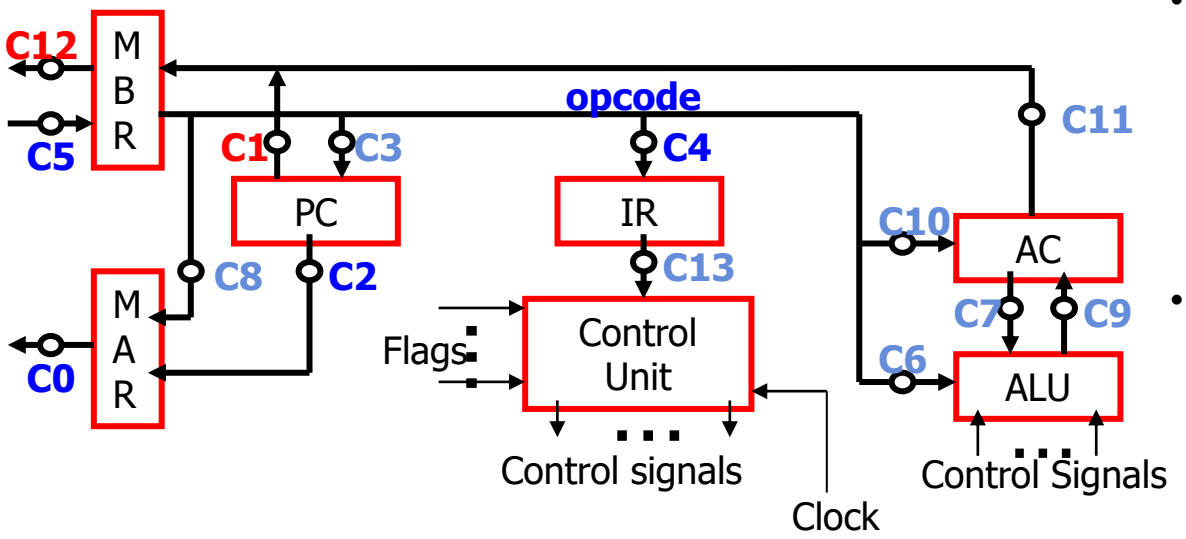
Indirect:

t1: MAR ← (IR(address)) C8,

t2: MBR ← Memory C0，C5, CR

t3: IR(address) ← (MBR(address)) C4

Interrupt cycle 中断周期



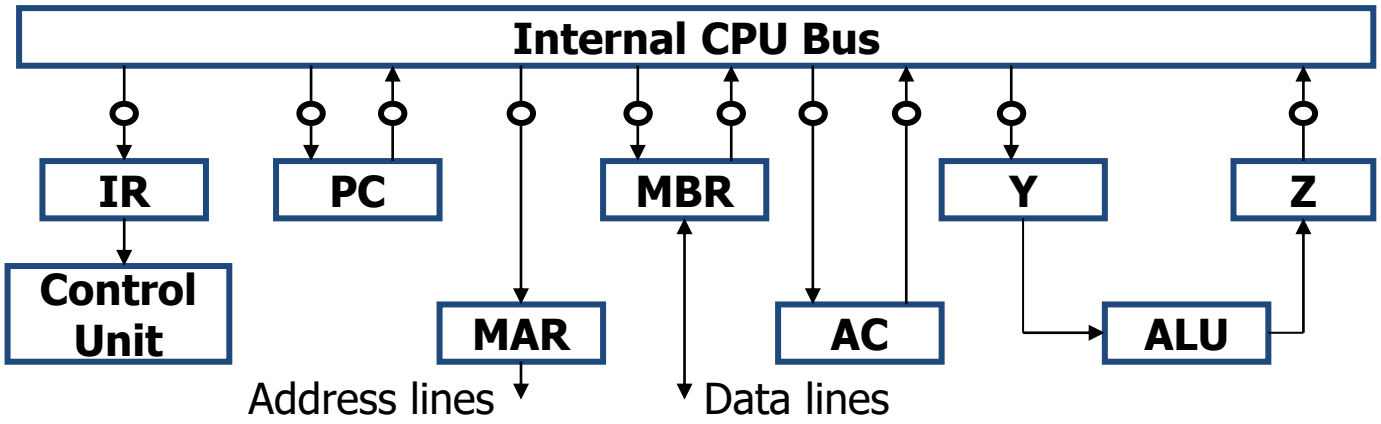
Interrupt:
t1: MBR ← (PC) C1
t2: MAR ← Save-address
PC ← Routine-address
t3: Memory ← (MBR) C12, CW

- T1周期：需要把PC的内容给MBR，以在中断结束后返回原来指令的执行，打开C1
- T2周期：需要保存PC的地址给MAR，同时把中断处理程序的起始地址给PC。这个控制信号没有在图上显示
- T3周期：MBR中的内容要存到存储器中，要打开C12，完成这个数据传送

Internal organization 内部组织

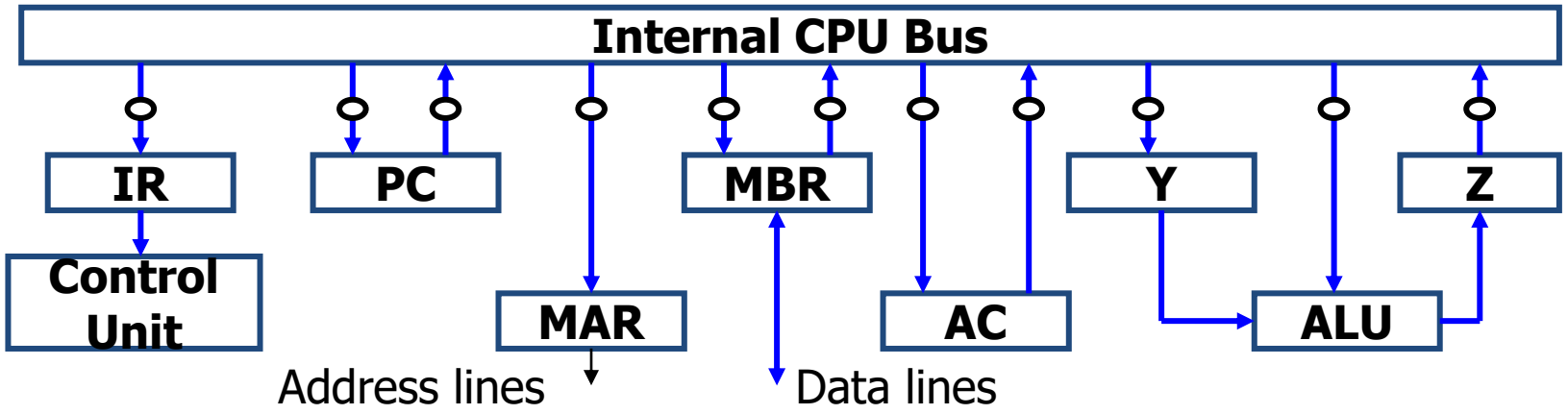
- CPU adopts single bus structure CPU内部采用单总线结构
  - All components, including ALU and registers, are connected to the bus 各个部件，包括ALU和寄存器，都连到总线
  - A gate is set between each component and the bus to control the data transmission between the component and the bus 各部件和总线之间设置门，控制部件和总线之间的数据传输
- Data transfer to and from external systems bus is controlled by the control signal 和外部系统总线之间的传输由控制信号来控制
- Temporary registers needed for proper operation of ALU 临时寄存器保证ALU正常运行

Internal processor organization 内部处理器组织



- 各个寄存器都链接到内部总线上。寄存器和内部总线之间有个逻辑门，用于寄存器和内部总线的连接控制
- ALU没有内部存储部件。如果ALU的操作涉及2个操作数的话，一个由内部总线得到，另一个必须通过其他的数据源来得到，不能都在内部总线上。需要增加一个Y寄存器，用于临时保存源操作数
- ALU还需要有一个寄存器，来临时保存输出结果。这个结果不能总线上，因为如果放总线上的话，它又会当作ALU的输入

Example



ADD X

- t1: MAR ← (IR(address))
- t2: MBR ← Memory
- t3: Y ← (MBR)
- t4: Z ← (AC) + (Y)
- t5: AC ← (Z)

- T1：IR中的操作数地址给MAR
- T2：存储器中数据给MBR
- T3：MBR中的源操作数给寄存器Y
- T4：ALU执行加法操作，把AC和Y中的内容相加，放到Z寄存器中
- T5：Z寄存器的内容保存在AC中

Implementation of control unit 控制器实现方法

- The control unit implementation can be done by two methods 控制器实现有两种方式
  - Hardwired implementation 硬布线实现
  - Microprogrammed implementation 微程序控制实现
- Hardwired implementation: the control unit is a combinational circuit 硬布线实现：控制单元是一个组合电路
  - The input signals are transferred into a set of output control signals 输入信号被转换成一组输出控制信号

Hardwired Implementation 硬布线实现

- Control Unit Inputs 控制器输入
  - Instruction register 指令寄存器
  - Flags 标志
  - Control bus 控制总线
  - Clock 时钟
- Control Unit Logic 控制器逻辑
  - Implement using discrete gates 使用离散门实现
  - Generate control signals for each gate 生成各个门的控制信号

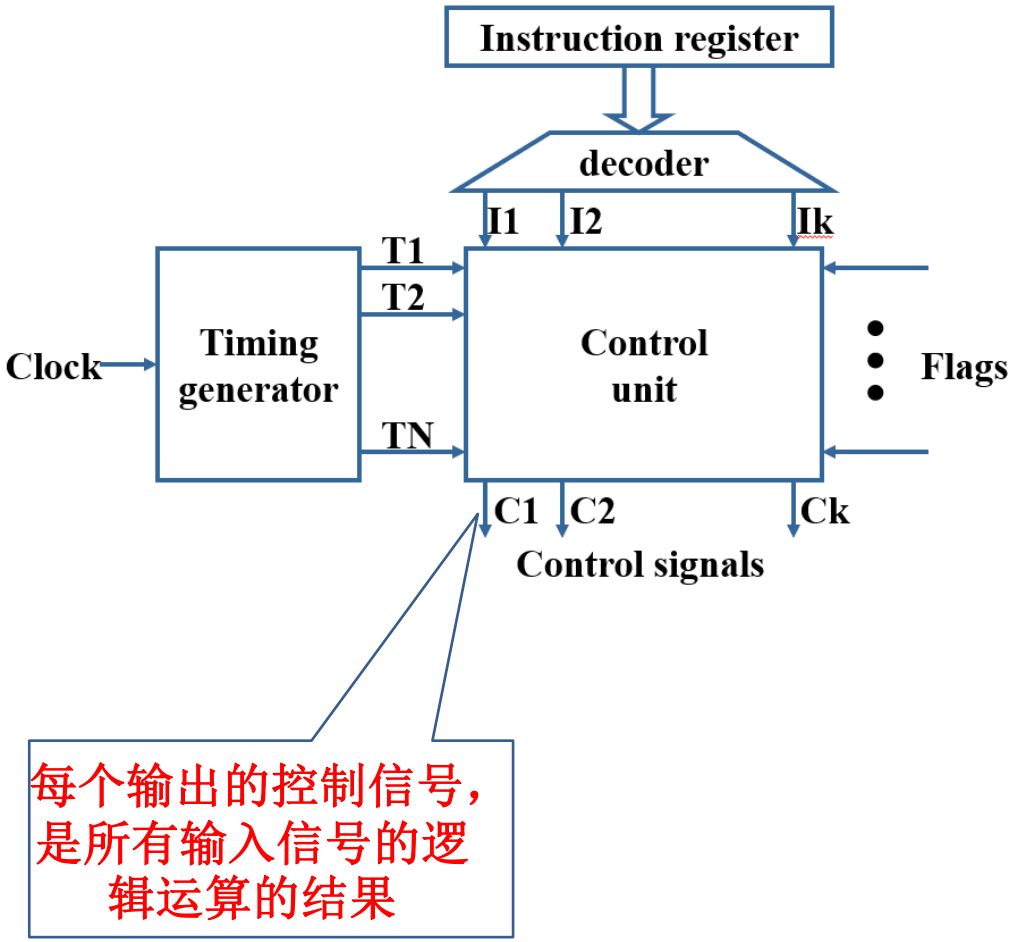
Control unit inputs 控制器输入1

- Flags and control bus signal 标志和控制总线信号
  - Bitwise processing 按位来处理
  - Each bit means something 每一位表示一个事件
- Instruction register 指令寄存器
  - Op-code causes different control signals for each different instruction 操作码为每个不同的指令产生不同的控制信号
  - Unique logic for each op-code 每个操作码一个唯一逻辑
  - Decoder takes encoded input and produces single output 译码器接收编码输入并产生单个输出
  - *n* binary inputs and 2^n outputs n个二进制输入和2^n个输出

Control unit inputs 控制器输入2

- Clock 时钟
  - Repetitive sequence of pulses 重复的脉冲序列
  - Useful for measuring duration of micro-ops 对于测量微操作的持续时间很有用
  - Must be long enough to allow signal propagation 必须足够长以允许信号传播
  - Different control signals at different times within instruction cycle 指令周期内不同时间有不同的控制信号
  - Need a counter with different control signals for t1, t2 etc. 对于周期中的步骤如t1、t2，需要有计数器，来产生不同控制信号

Control unit logic 控制器逻辑



- 时钟发生器：产生时钟信号，并提供计数器
- 译码器：根据指令寄存器中的指令操作码，生成操作码对应的唯一输出，提供给输出控制单元
- 标志：为控制单元提供相关的标志信号
- 控制单元：是控制器的核心，负责产生控制信号

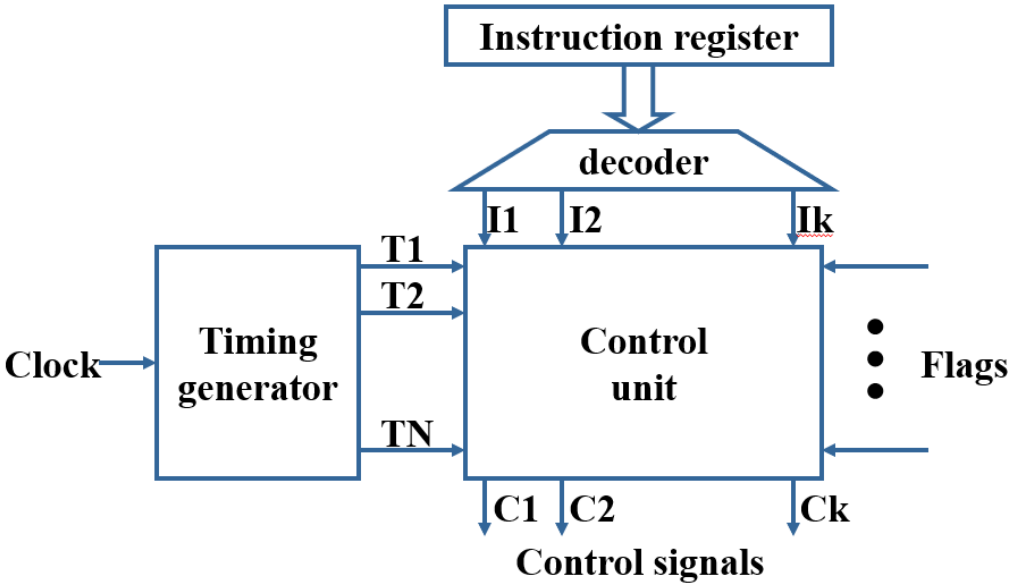
Control matrix 控制矩阵

- A gate is set between each component and the bus to control the data transmission between the component and the bus 各部件和总线之间设置门，控制部件和总线之间的数据传输
- The purpose of the control unit is to generate control signals and control the opening and closing of each door 控制单元的目的就是生成控制信号，控制各个门的开关
  - Design a Boolean expression composed of input signals for each control signal 为每一个控制信号设计一个由输入信号组成的布尔表达式
  - Determine the door switch according to the calculation result of Boolean expression 根据布尔表达式的计算结果，确定门的开关

Control matrix 控制矩阵

- There are many input signals related to each gate 和每个门相关的输入信号很多
- The Boolean expression for each door is complex 每个门的布尔表达式很复杂
- Usually done using PLA（Programmable logic array ）通常使用PLA（可编程逻辑阵列）完成
  - Programmable arrays may also be numerous 可编程阵列也可能会很多
  - Large control matrices are implemented hierarchically for speed为了提高速度，大型控制矩阵分层实现

Example



- 比如，C1和I1，I2，In，T2，有关，那么C1需要通过这几个输入生成一个布尔表达式，来表示这几个输入在什么状态的时候，C1打开
- 输入的指令种类多，还有多个时钟周期，以及标志位等等，控制信号的布尔表达式可能会非常复杂
- 当某个输入发生变化的时候，还需要调整所有跟这个输入相关的控制信号的布尔式

Problems With Hard Wired 硬布线问题1

- Complex micro-operation logic 复杂的排序与微操作逻辑
  - CPU functions are very complex, and there are many kinds of micro operations CPU功能非常复杂，微操作种类多
  - The control signal of each door is very complex 每个门的控制信号非常复杂
  - The design and implementation of such complex Boolean expressions, sorting and logic are very complex 设计和实现这样复杂的布尔表达式，排序和逻辑都很复杂
- Difficult to design and test 难以设计和测试
  - Difficulty in Circuit Design of Boolean Expressions 布尔表达式的电路设计困难
  - The complexity of testing is also high 测试的复杂度也很高

Problems With Hard Wired 硬布线问题2

- Inflexible design 设计不灵活
  - Design Boolean expression of each control signal according to micro operation set 按照微操作集设计每个控制信号的布尔表达式
  - If there are changes, all relevant Boolean expressions need to be changed and redesigned 如果有变动，所有相关的布尔表达式都需要变化，需要重新设计
- Difficult to add new instructions 难以添加新指令
  - Add a new instruction, and all control signals related to the instruction need to be redesigned and sequenced 添加一个新指令，和指令相关的所有控制信号都需要重新设计和排序
  - Very heavy workload 工作量非常大

How?

- Hard wiring is designed from the perspective of control door

  硬布线是从控制门的角度去设计

  - Which input signals are relevant to this gate 哪些输入信号跟门相关
- Another idea is to analyze the control gates related to each micro operation from the perspective of micro operation 换个思路，从微操作的角度考虑，分析每个微操作相关的控制门
- That is another well known process and design method： Microprogrammed Control 这就是另一种更常用的处理和设计方法：微程序控制

Summary1 小结1

- The instructions are divided into a sequence of stages 指令分为一系列阶段
  - Fetch, indirect, execute, interrupt 取指，间接，执行，中断
- Each stage is further divided into a sequence of micro-operations 每个阶段进一步划分为一系列微操作
  - T1, T2, T3, …
- Each micro-operation occupies one clock unit 每个微操作占用一个时钟单元

Summary2 小结2

- The function of the control unit is to produce a sequence of timed control signals to control the data paths and ALU operations 控制单元的功能是产生一系列定时控制信号，以控制数据流动和ALU操作
- The control unit is a combinational circuit that transfers inputs to a group of control signals 控制单元是一个组合电路，将输入信号转换成一组控制信号
  - Inputs: flags, instruction registers, clock, control signals from system bus 输入：标志、指令寄存器、时钟、系统总线的控制信号
  - Outputs: control signals to system bus and within CPU 输出：至系统总线和CPU内的控制信号

Outline

- Control Unit Operation
  - Micro-Operations 微操作
  - Control of the Processor 处理器控制
  - Hardwired Implementation 硬布线实现方式
- Microprogrammed Control
  - Basic Concepts 基本概念
  - Microinstruction Sequencing 微指令定序
  - Microinstruction Execution 微指令执行

History 1

- The idea of hardwired system is relatively simple, but the design is complex and inflexible 硬布线系统思路比较简单，但是设计复杂、不灵活
- The concept of microprogram was proposed by Wikes in the 1950s 微程序的概念是20世纪50年代，由Wikes提出来的
  - A Controller Design Method Avoiding the Complexity of Hard wired Implementation 一种避免了硬布线实现复杂性的控制器设计方法
  - A fast and inexpensive control memory is required 要求一个快速并且不太昂贵的控制存储器
  - It was not widely used at that time 当时并没有得到广泛使用

History 2

- In April 1964, IBM adopted microprogram design in most of its 360 series processors 1964年4月份，IBM在他的360系列处理器中大部分采用了微程序的设计
- Since then, microprogramming has become a popular technology for CISC processors 自那以后微程序设计成为CISC处理器的流行技术
- However, RISC architecture generally adopts hard wiring technology due to its regular and concise instructions 但是RISC架构由于指令规整、简洁，一般都采用硬布线技术

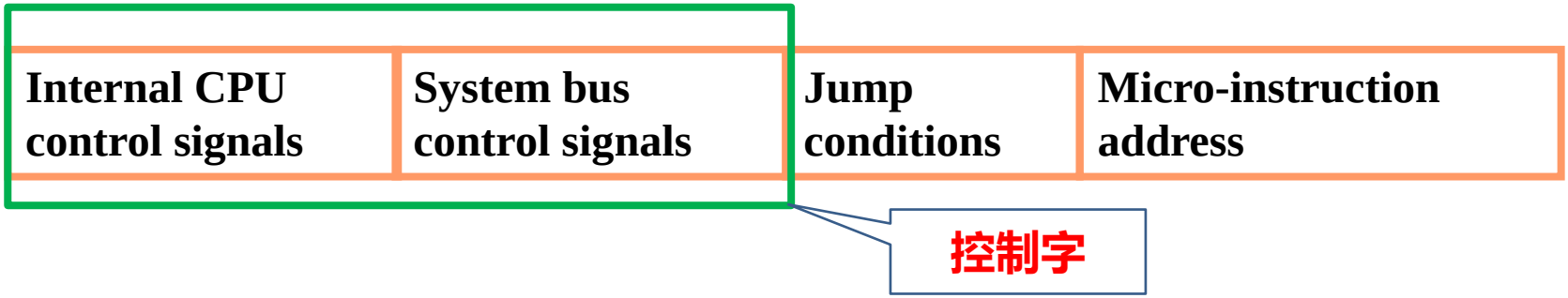
History 3

- In modern times, microprogramming is rarely used, but it is still a useful tool for computer design 到了现代，微程序设计已经很少用了，但还是计算机设计的有用工具
- For example
  - Pentium4 machine instructions are converted into RISC like forms Pentium4的机器指令转换成类RISC形式
  - most of them are no longer executed by microprograms 大部分已经不用微程序来执行
  - But a few of them are still executed by microprograms 但是有少量指令还是用微程序

Basic Concepts 基本概念

- Control unit can be implemented by a more flexible technique called microprogrammed control unit 控制单元可以通过一种称为微程序控制单元的更灵活的技术来实现
  - The logic of the control unit is specified by a microprogram 控制单元的逻辑由微程序描述
  - Each line in microprogram describes a group of micro-operations in one time，called microinstruction 微程序中的每行描述一个时间内出现的一组微操作，称为微指令
  - Construct a control word corresponding to this microinstruction 对应这个微指令，构造一个控制字
  - The control word determines the opening or closing of all doors 控制字决定所有门的开或者关

Microinstruction 微指令

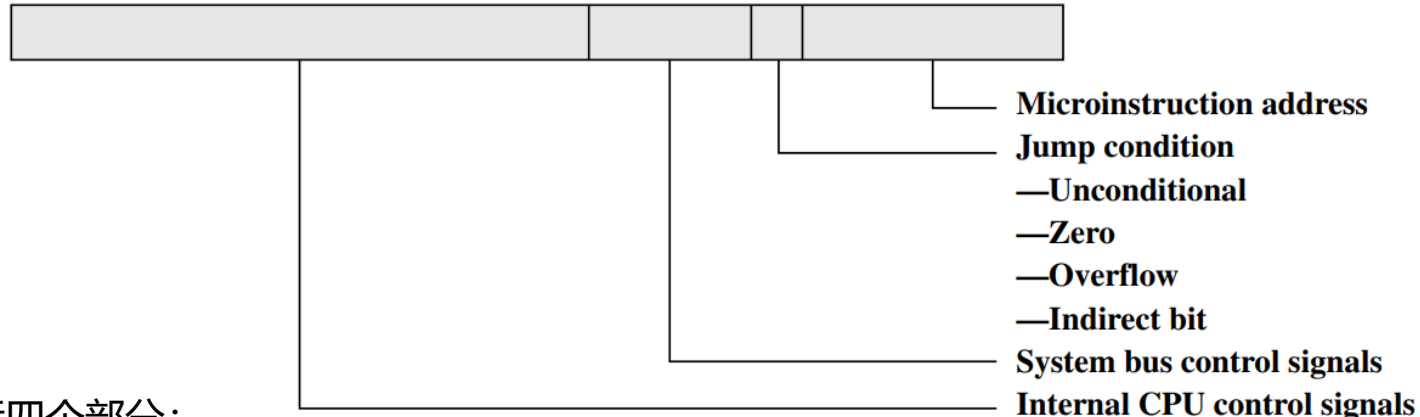


- 微指令中包括CPU内控制信号、系统总线控制信号、跳转条件、微指令地址
- 执行这条微指令的效果是，打开CPU内外所有为1的控制线，关闭所有为0的控制线
- 如果条件条件为假，自动执行下一个顺序的微指令
- 如果跳转条件为真，则执行“微指令地址”中对应的微指令

Micro-instruction Types 微指令类型

- horizontal microinstruction 水平型微指令
- vertical microinstruction 垂直型微指令

Horizontal Micro-programming 水平微指令

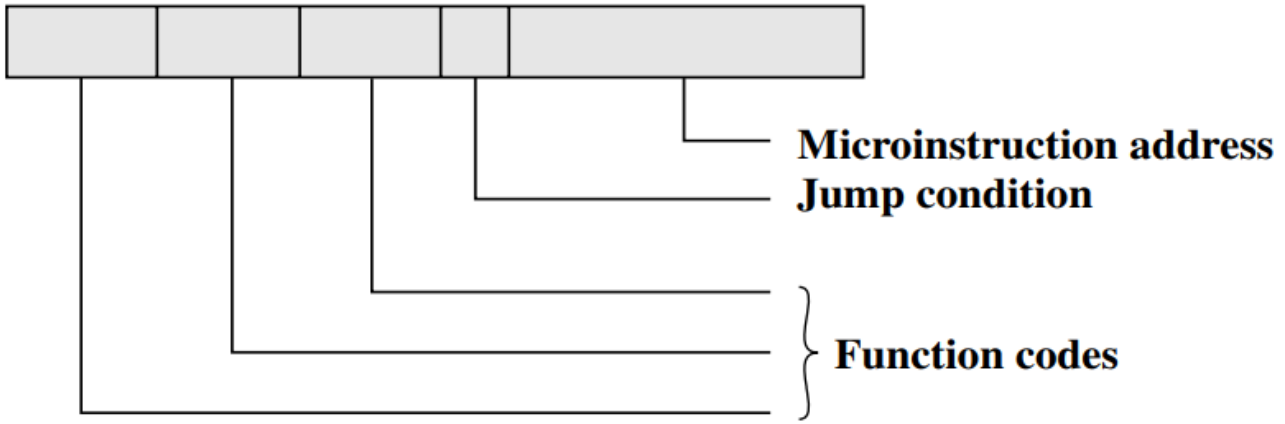


- 水平微指令包括四个部分：
  - 内部控制信号：CPU内的每一个控制线都有相应的1位
  - 总线控制信号：每一个系统控制总线都有相应的1位
  - 条件字段：指示转移发生条件的字段
  - 微指令地址：转移的目标指令地址
- 打开位值为1的控制线，关闭所有位值为0的控制线。执行一个或多个微操作
- 如果跳转条件为假，顺序执行下一个指令。如果跳转条件为真，执行“微指令地址” 指向的微指令

Characteristic 特点

- Each control line inside the CPU and bus needs to have a separate control bit CPU内部和总线的每一个控制线都需要有一个单独的控制位
  - The width of the control word is very large 控制字要求的字宽非常大
- Each line is a separate control bit 每个线都是单独的控制位
  - Each control line can be controlled independently 各个控制线的控制可以独立进行
  - More kinds of micro operations that can be supported 可支持的微操作种类多
- The control lines are controlled by one bit of the control word independently 控制线都是由控制字中的1位单独控制
  - The coding of control signal is not compact 控制信号的编码不紧凑

Vertical Micro-programming Formats 垂直微指令



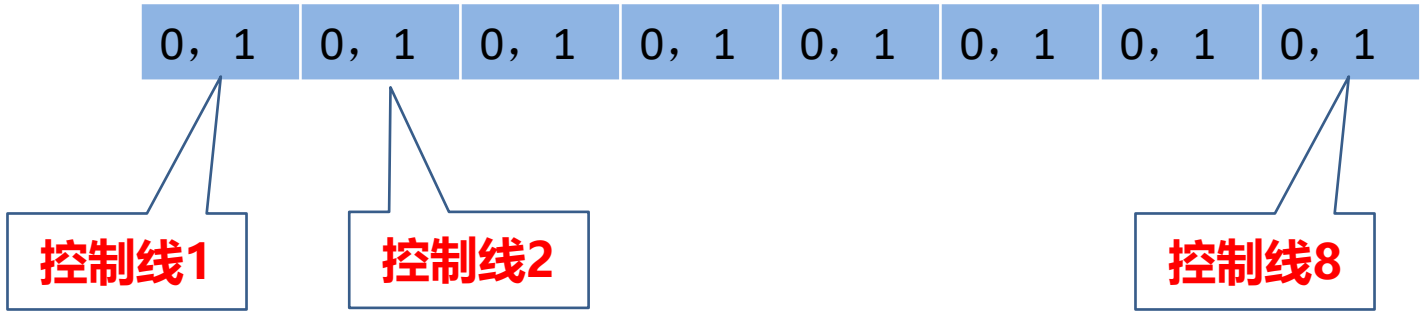
- 垂直微指令也包括跳转条件和跳转微指令地址，跟水平微指令类似
- 在控制方面，和水平微指令的每一位对应一个控制线不同，垂直微指令又做了一次编码
- 解决水平微指令的控制字太长的问题
- 编码后，由一个解码器再翻译成控制线的控制信号。

Characteristic 特点

- Width is narrow 微指令宽度窄
  - n control signals encoded into $\log_2 n$ bits n位控制字最多可以表示2^n种控制信号组合
  - Limited number of micro operations supported 支持的微操作数量有限
- Additional steps required 需要额外的步骤
  - External memory word decode is needed 需要外部存储器中的字解码器
  - Identify the exact control line being manipulated 精确识别被操纵的控制线
  - Send control signal 发出控制信号

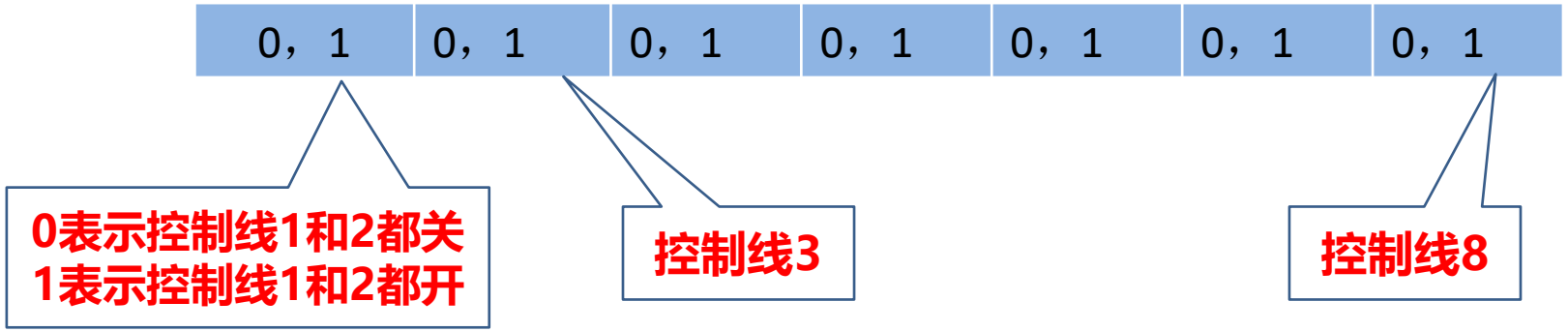
Example

- 水平微指令
  - 8根控制线，每一根都可以独立控制。最多可以表示2^8=256种组合，实现2^8=256种控制
  - 假如在微指令级别上，控制线1和2必须都同时开或同时关。那么控制字中，01XXXXXX和10XXXXXX不会出现。这就存在控制字位的浪费情况



Example

- 垂直微指令
  - 采用垂直微指令，可以将控制线1和2重新编码。0表示控制线1和2都关闭，1表示控制线1和2都开。
  - 这样，可以用七位来进行控制。0XXXXXX表示关闭控制线1和2，1XXXXXX表示打开控制线1和2



Micro-program Word Length 微程序字长

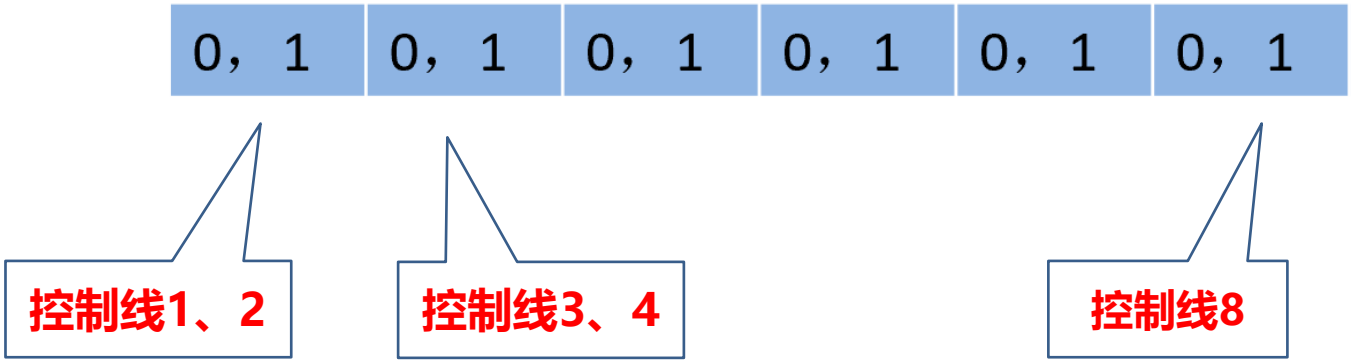
- Word length is an important factor in microprogram微程序的字长是微程序中的重要因素
- Word length is determined by three factors 微程序的字长由三个因素决定
  - Maximum number of simultaneous micro-operations supported 所支持的最大微操作类型
  - The way control information is represented or encoded 控制信息表示及编码方式
  - The way in which the next micro-instruction address is specified 指定下一个微指令地址的方式

Compromise 折中方案

- Divide control signals into disjoint groups 将控制信号分成不相交的互斥组
- Implement each group as separate field in memory word 在存储器的控制字中，每个组实现为单独的字段
- Supports reasonable levels of parallelism without too much complexity 支持合理级别的并行性，而不会太复杂

Example

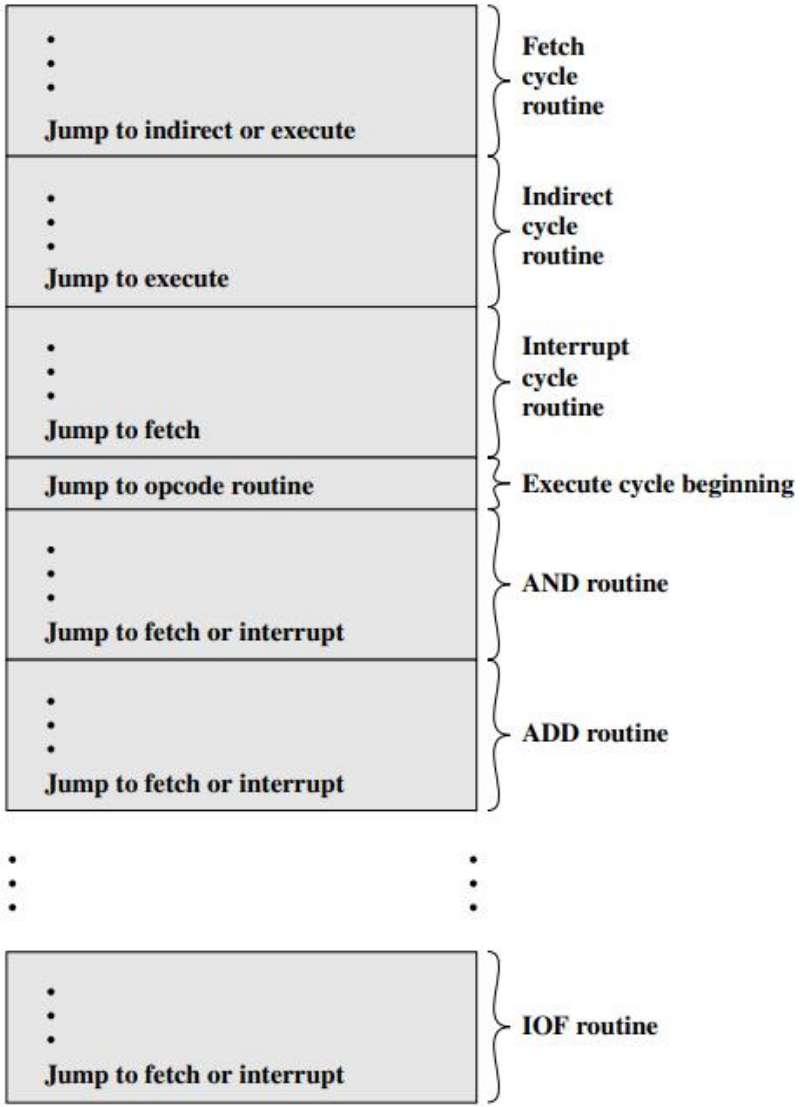
- 如果控制线1~4均为同时开或者关，其他的都独立控制，那么最少只需要5位控制字
  - 0XXXX表示关闭控制线1~4，1XXXX表示打开控制线1~4
  - 此时控制线1~4都只能同时开和关，没有并行性
- 可以将控制线1和2分为一组，控制线3和4分为一组。这样需要6位控制字
  - ABXXXX。AB为00表示1~4都关，01表示1~2关，3~4开，10表示1~2开，3-4关，11表示·1~4都开



The control memory 控制存储器

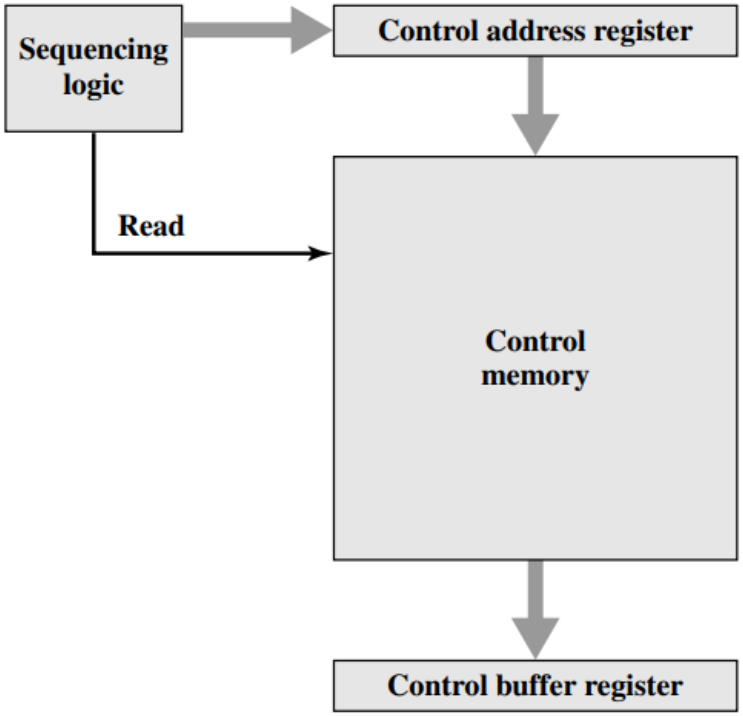
- The control words are put into a special memory block called control memory, with each word having a unique address 控制字被放入一个称为控制存储器的特殊内存块中，每个字都有一个唯一的地址
- The microinstructions are organized as different routines 微指令被组织为不同的例程
  - The microinstructions in each routine are executed sequentially 每个例程中的微指令按顺序执行
  - Each routine ends with a branch instruction points to the next routine 每个例程结束的地方，有一个分支，指向下一个例程

Control memory 控制存储器



- 存储控制器中存储了取指例程、间接例程、中断例程、执行例程等，这些例程规定了在每个周期内需要执行的微操作序列。
- 例程中还规定了在执行结束后转移到下面哪一个例程，也就是指定了周期的执行顺序。
- 例如，取指周期后，可能会到间接周期，也可能到执行周期，需要根据取指周期来确定

Control Unit Microarchitecture 控制器微结构



- 控制存储器：存储微指令
- 控制地址寄存器：包含下一个将要被读取的微指令地址。类似于PC
- 控制缓冲寄存器：微指令由控制存储器读取之后，放到控制缓冲寄存器中。控制缓冲寄存器的左半部分直接和控制线连接，读取微指令直接生成控制信号，控制门的开或关
- 定序逻辑：负责为控制地址寄存器提供地址，并发出读命令

Sequencing logic 定序逻辑

Control address register 控制地址寄存器

Control memory 控制存储器

Control buffer register 控制缓冲寄存器

Control Unit Function 控制器功能

- Sequence logic unit issues read command 定序逻辑单元发出读取命令
  - Word specified in control address register is read into control buffer register 控制地址寄存器中指定的字被读入控制缓冲寄存器
  - Control buffer register contents generates control signals and next address information 控制缓冲寄存器内容生成控制信号和下一个地址信息
- Sequence logic loads new address into control address register based on next address information from control buffer register and ALU flags 定序逻辑基于来自控制缓冲寄存器和ALU标志生成下一个地址信息，将新地址加载到控制地址寄存器中

Next Address Decision 下一个地址的决定

- Sequencing logic needs to determine the next microinstruction

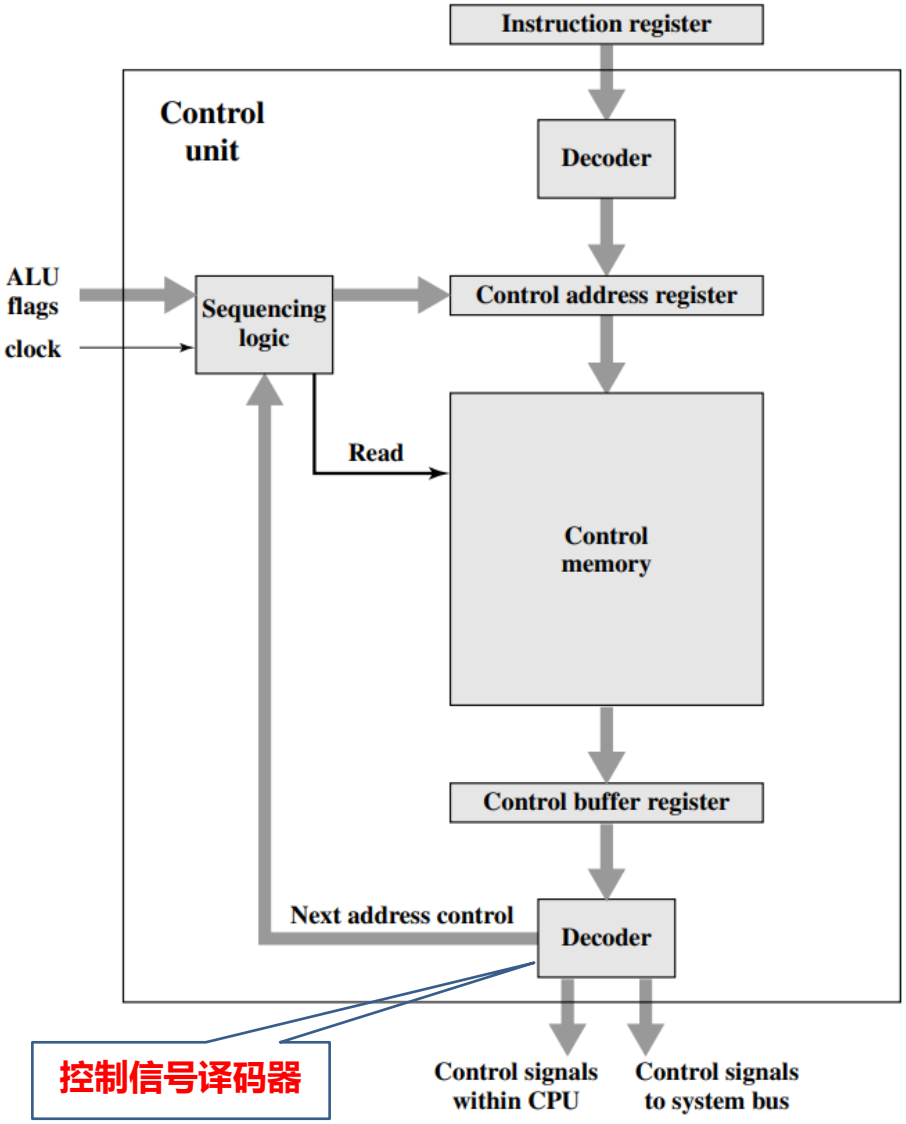
  定序逻辑需要确定下一个微指令

  - At the end of each microinstruction, the next microinstruction address is loaded into the control address register 在每条微指令结束时，把下一个指令地址装入到控制地址寄存器中
- Depending on ALU flags and control buffer register 基于ALU标志和控制缓冲寄存器
  - Next microinstruction in sequence 顺序的下一个微指令
  - Jump to a new routine based on jump instructions 基于跳转指令转移到新的例程
  - Jump to a machine instruction routine转移到一个机器指令例程

Next Address 下一个地址

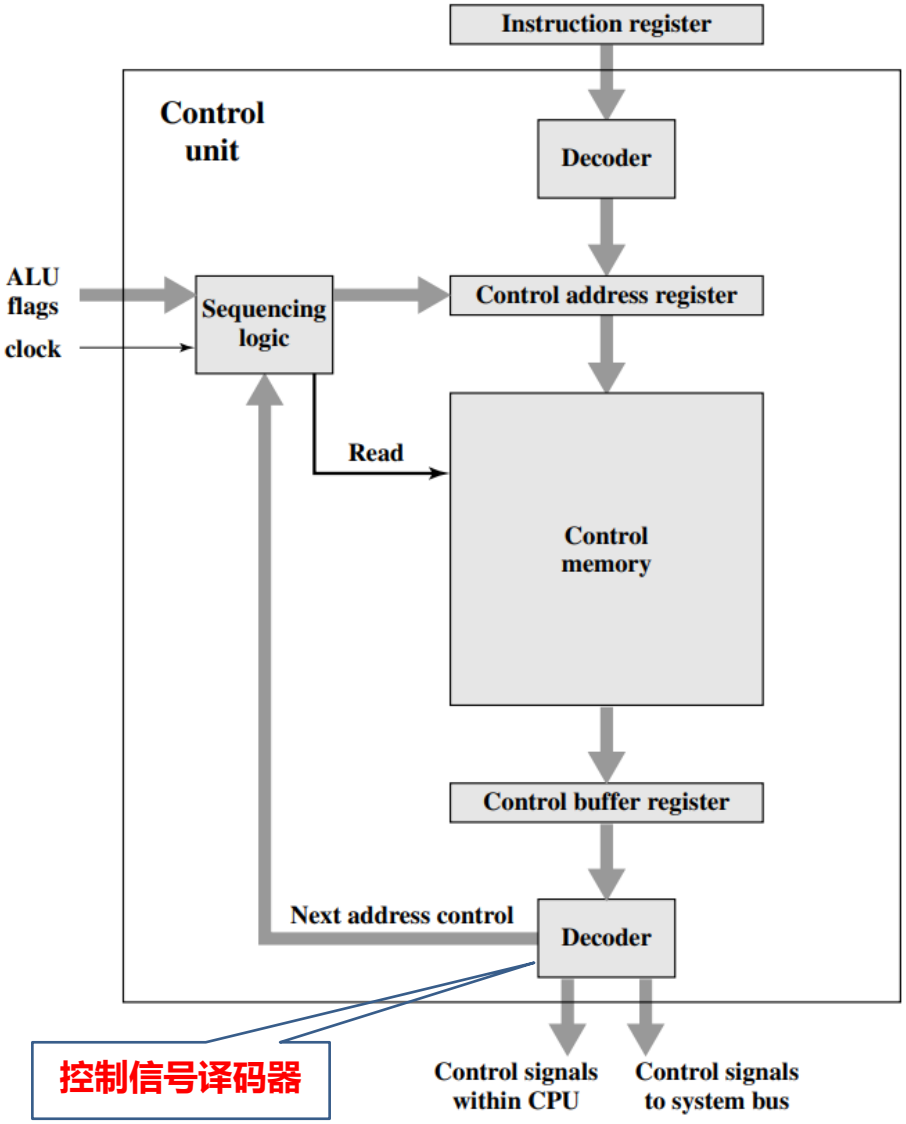
- Next microinstruction in sequence 下一个顺序微指令
  - Add 1 to control address register 在控制地址寄存器上+1
- Jump to new routine based on jump microinstruction 基于跳转微指令跳到新的例程
  - Load address field of control buffer register into control address register 将控制缓冲寄存器的地址字段加载到控制地址寄存器中
- Jump to machine instruction routine 跳到机器指令例程
  - Load control address register based on opcode in IR 基于IR中的操作码来加载控制地址寄存器

Functioning of Control Unit 微程序控制器的运作1



1. 指令译码器是将指令寄存器中的操作码翻译成控制寄存器地址
2. 控制地址寄存器到控制存储器中找到控制字，并放到控制缓冲寄存器中。
3. 控制缓冲寄存器进行控制操作。如果是垂直微指令，则需要译码器进行译码后生成控制信号

Functioning of Control Unit 微程序控制器的运作2



4. 控制信号译码器用于垂直微指令，将控制字翻译成控制线的控制信号，发送到CPU内部和系统总线。
5. 对于垂直微指令，有一个附加逻辑和时间延迟
6. 下一个微指令地址给定序逻辑，以确定执行哪一个微指令

Simple summary -1

- All the control unit does is generate a set of control signals 控制器的工作就是生成一组控制信号
  - Control the opening and closing of each gate 控制各个门的开与关
  - Use one control bit to represent the control signal of a door 用一位控制位表示一个门的控制信号
  - Control bits of all gates constitute control words 所有门的控制位组成控制字
- Have a control word for each group of micro-operations in one time 每组同时进行的微操作都有一个控制字
- The next microinstruction can be specified in the microinstruction and executed conditionally 可以在微指令中指定下一条微指令，并根据条件执行

Simple summary -2

- For machine code instruction 对于机器代码指令
  - It is generally divided into multiple cycles, such as fetch cycle, execution cycle, etc. 一般分为多个周期，如取指周期、执行周期等
  - Multiple microoperations need to be completed 每个周期需要完成多个微操作
  - Each microoperation corresponds to one control word 每个微操作对应1个控制字
  - A group of control words realizes the control function of a machine code 一组控制字实现一个机器代码的控制功能

Simple summary -3

- Today’ s large microprocessor 今天的大型微处理器
  - Many instructions and associated register-level hardware 许多指令和相关的寄存器级硬件
  - Many control points to be manipulated 许多控制点需要操作
- This results in control memory that 这将导致控制存储器
  - Contains a large number of words，co-responding to the number of instructions to be executed 包含大量的控制字，共同响应要执行的大量指令
  - Has a wide word width，due to the large number of control points to be manipulated 由于需要操纵的控制点数量众多，所以具有较宽的字宽

Advantages and Disadvantages 优缺点

- Advantages
  - Simplifies design of control unit 简化控制器设计
  - Cheaper 便宜
  - Less error-prone 不易出错
- Disadvantages
  - Additional processing required 需要额外的处理
  - Slower 慢

Outline

- Control Unit Operation
  - Micro-Operations 微操作
  - Control of the Processor 处理器控制
  - Hardwired Implementation 硬布线实现方式
- Microprogrammed Control
  - Basic Concepts 基本概念
  - Microinstruction Sequencing 微指令定序
  - Microinstruction Execution 微指令执行

Main Tasks 主要任务

- Microinstruction controller has two basic tasks 微程序控制器有2个基本任务
  - Microinstruction sequencing： Get the next microinstruction from the control memory 微指令定序：由控制存储器得到下一条微指令
  - Microinstruction execution : Through control word generates control signal for executing the micro instruction 微指令执行：通过控制字产生执行微指令的控制信号
- Must consider both together 两者一起考虑
  - Affects instruction format and controller timing 影响指令的格式和控制器的时序

Sequencing Techniques 定序技术

- Two problems must be considered 需要考虑2个问题
  - Size of microinstructions 微指令的大小
  - Address generation time 地址生成的时间
- Size of microinstructions 微指令大小
  - The larger the microinstruction, the larger the control memory, and the more expensive 微指令越大，控制存储器就越大，价格就越贵
  - Size of microinstructions should be reduced 应减小微指令的大小
- Address generation time 地址生成时间
  - The requirement to execute microinstructions as quickly as possible 尽可能快地执行微指令的要求

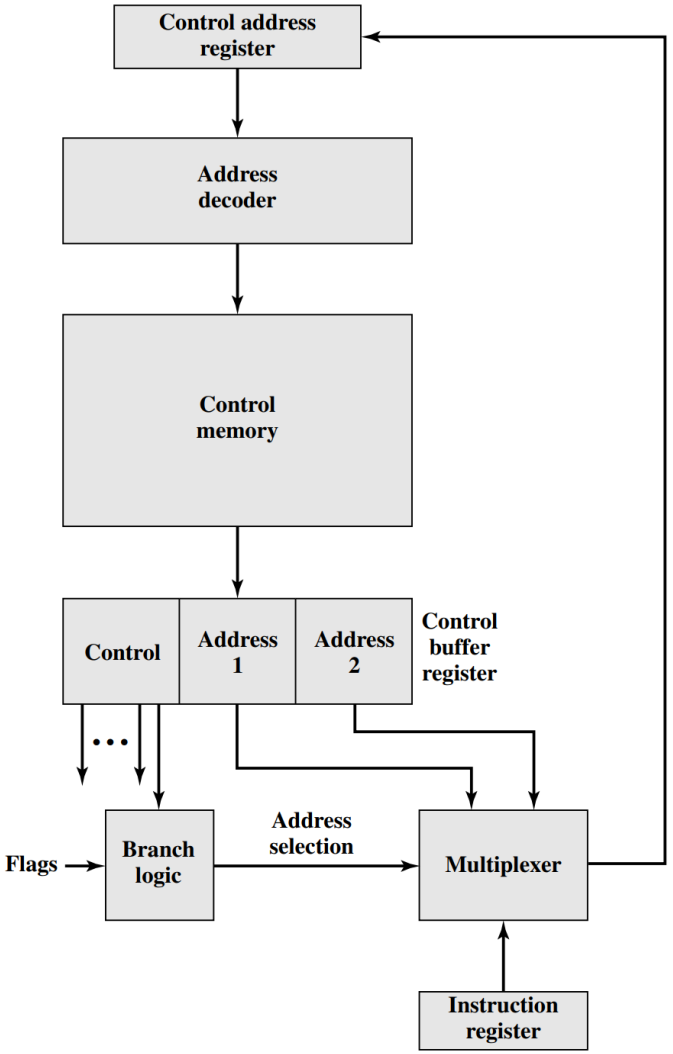
Address generation time 地址生成时间

- Determined by instruction register 指令寄存器决定
  - Once per cycle, after instruction is fetched 每个周期一次，在取指结束后
- Next sequential address 下一个顺序地址
  - Use in most cases 大多数情况下使用
- Branche 分支
  - Both conditional and unconditional 条件或无条件
  - Generally, there is a jump after 3-4 microinstructions 一般3-4条微指令后就有一个转移
  - Very important 非常重要

Sequencing Techniques 定序技术

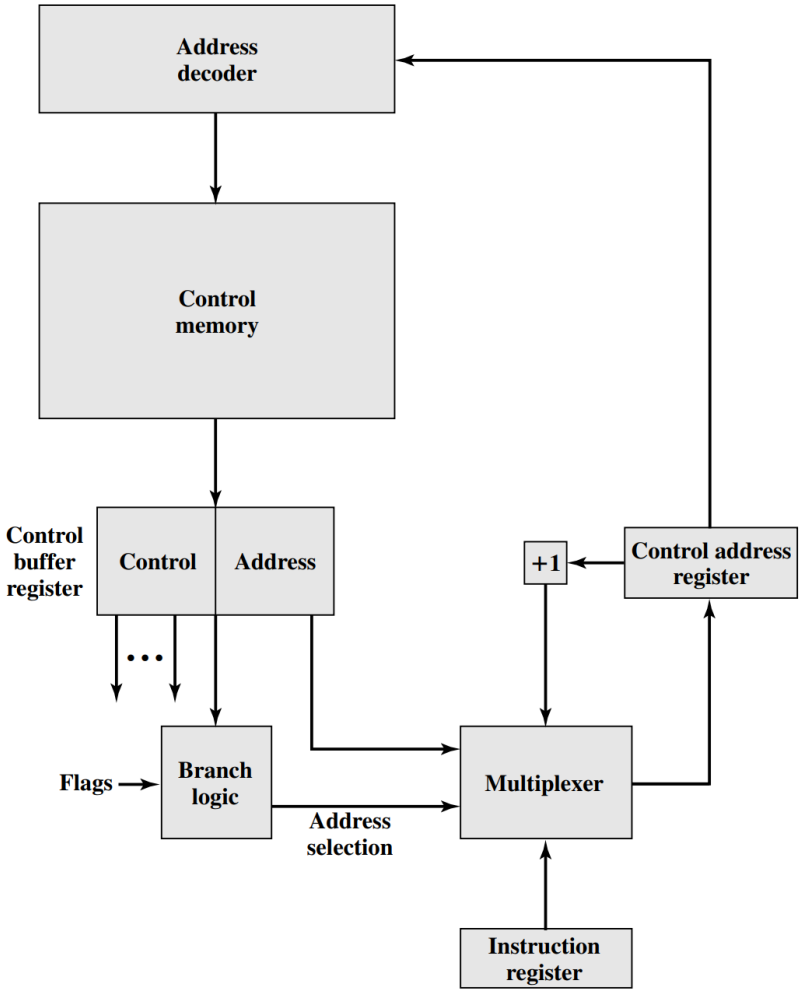
- Based on current microinstruction, condition flags, contents of IR, generate control memory address of next microinstruction
  基于当前微指令，条件标志，IR的内容, 生成下一个微指令在控制寄存器中的地址
- The jump of microinstructions can be divided into three modes according to the address information format in the microinstructions 对于微指令的转移，根据微指令中的地址信息格式分为三种方式
  - Two address fields 双地址字段
  - Single address field 单地址字段
  - Variable format 可变格式

Two address fields 双地址字段



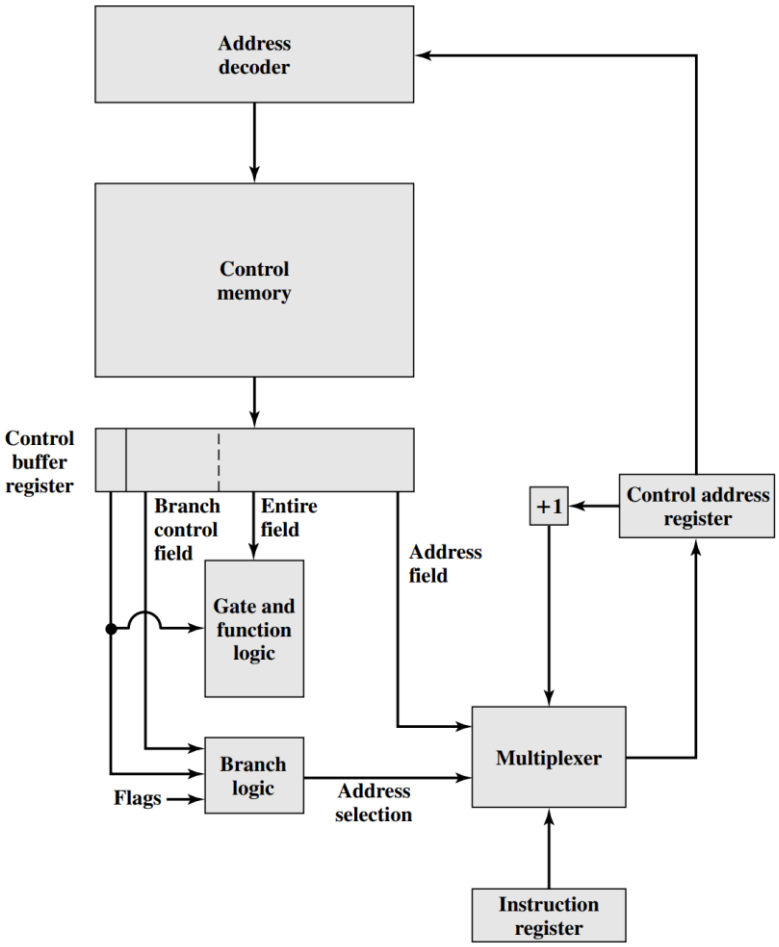
- 对于双地址字段模式，微指令中提供了2个地址字段
- 控制缓冲寄存器读取微指令后，第一部分产生控制信号，第二部分和第三部分是2个地址，这2个地址，加上指令寄存器的内容，连接到一个多路器中。多路器选择一个送到控制地址寄存器中
- 控制地址寄存器通过译码，产生下一个微指令的地址。地址选择的输入是由转移逻辑模块来完成的，它根据微指令中的控制部分，以及标志位，向多路器提供选择信号
- 双地址字段方式比较简单，微指令中有2个地址，需要更多的位

Single address fields 单地址字段



- 单地址字段方式中，微指令中只有一个地址。下一个地址的选择方法是：
  - 地址字段中指定的地址
  - 指令寄存器中的代码
  - 下一个顺序地址
- 下一个顺序地址是用上一次的控制地址寄存器中的地址+1来得到的

Variable format 可变格式



- 可变格式提供两种不同格式的指令。用1位来标识当前是哪种格式。
- 在一种格式中，所有位都用于产生控制信号。所以，下一个微指令的地址，或者是下一顺序地址，或者是由指令寄存器来指定
- 另一种格式中，有一些位用于启动转移逻辑模块，其余的位来提供地址。这样就可以进行条件或者无条件转移
- 这种方法的缺点是转移微指令需要花费一个时钟周期。

Outline

- Control Unit Operation
  - Micro-Operations 微操作
  - Control of the Processor 处理器控制
  - Hardwired Implementation 硬布线实现方式
- Microprogrammed Control
  - Basic Concepts 基本概念
  - Microinstruction Sequencing 微指令定序
  - Microinstruction Execution 微指令执行

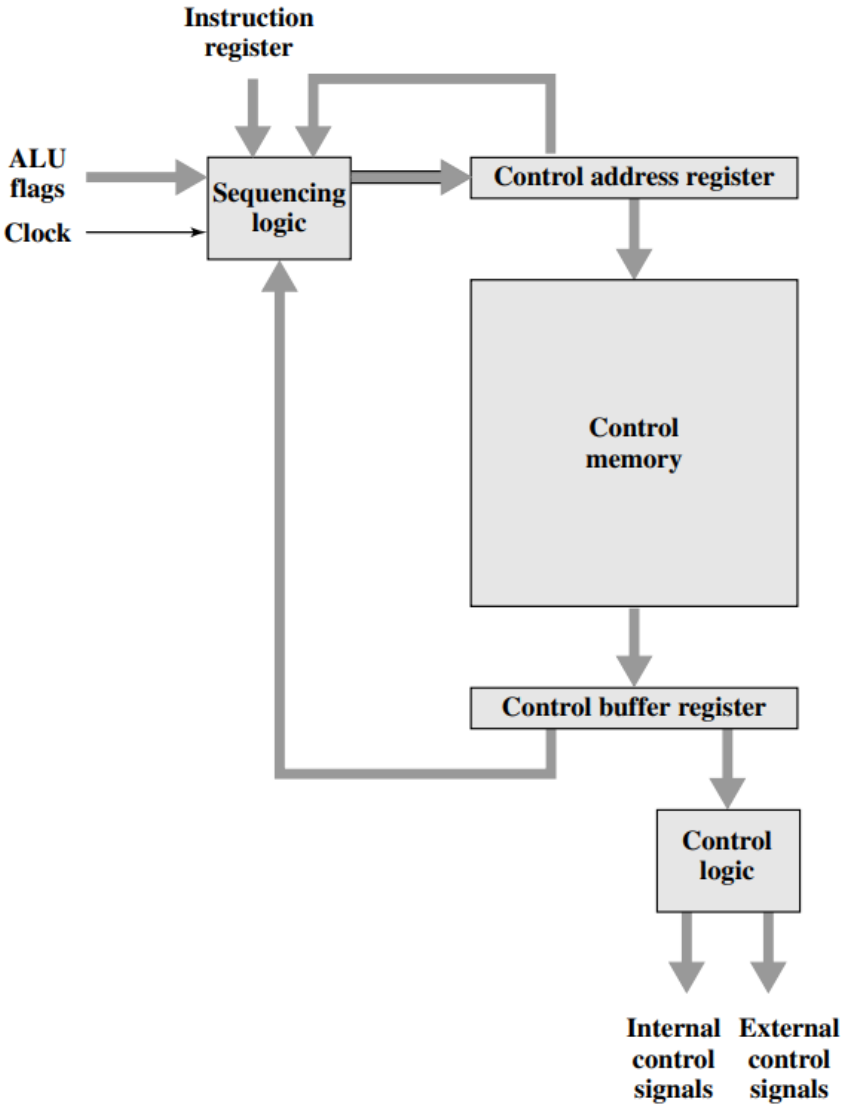
Microinstruction Execution 微指令执行

- Microinstruction execution is a periodic basic even 微指令执行是一个周期性的基本事件
- Each cycle is made up of two events 每个周期包括2个事情
  - Fetch 读取微指令
  - Execute 执行微指令
- Fetch
  - Get the microinstruction from the control memory according to the generated microinstruction address 根据生成的微指令地址，从控制存储器取得微指令
- Execute
  - Execute microinstructions and generate control signals 执行微指令，产生控制信号

Execution effect 执行效果

- Effect is to generate control signals 作用是产生控制信号
- Internal
  - Control signals inside the processor 处理器内部的控制信号
  - Control the internal components of the processor 对处理器内部各个部件进行控制
- External
  - Control signals to external control bus or other interface 发到控制总线或者其他接口的控制信号
  - Control disk read-write, I/O read-write etc. 控制磁盘读写，I/O读写等
- With the function to generate next address 附带功能，产生下一个地址

Control Unit Organization 控制器的组织



- 定序逻辑根据指令寄存器、ALU标记、时钟信号、控制地址寄存器、控制缓冲寄存器的内容，作为输入，产生下一个微指令的内容，发送给控制地址寄存器
- 控制存储器根据地址寄存器的内容，将微指令放到控制缓冲寄存器中
- 控制缓冲寄存器的一部分内容给定序逻辑，用于确定下一个微指令的地址。另一部分给控制逻辑，生成控制信号。
- 控制逻辑的复杂程度，和指令的格式以及内容有关系

Classification of microinstructions 微指令的分类

- Classification of microinstructions 微指令的分类方法
  - Vertical/horizontal 垂直/水平
  - Packed/unpacked 压缩/非压缩
  - Hard/soft microprogramming 硬/软微程序设计
  - Direct/indirect encoding 直接/间接编码

How to Encode -1

- Microinstructions are ultimately translated into control signals through control logic 微指令最终都要通过控制逻辑翻译成控制信号
- How many bits in microinstructions are appropriate? 采用多少位微指令合适呢？
- Assume that a total of k different internal and external control signals are required 假定总共需要k个内外控制信号
- First encode method
  - K bits，each is dedicated 用k位进行控制，每位都是专用的
  - 2^K control signals during any instruction cycle 任何指令周期有2^K种信号组合
- Is this method suitable? 这种编码方式是否合适？

How to Encode -2

- 2^K control signals，not all are used 2^K种信号组合中，不是所有的都用到
  - Two sources cannot be gated to same destination 2个源不能到同一个目的
  - Register cannot be source and destination 寄存器不能既是源，又是目的
  - Only one pattern presented to ALU at a time ALU同一个时刻只有一个模式
  - Only one pattern presented to external control bus at a time 同一时刻只能给外部控制总线提交一种控制信号样式
- K-bit control word wastes space k位控制字浪费了空间

How to Encode -3

- Second encode method
  - If there are Q types of control signal combination 假如控制信号组合有Q种
  - Only if the control bit k meets 2^K>Q 只需要控制位数k符合2^K >Q即可
- Disadvantage：coding is too compact 缺点：编码太紧凑
  - Difficult programming and poor concurrency 编程困难，并发性差
  - Requires complex slow control logic module 要求更复杂的控制逻辑，而且速度很慢

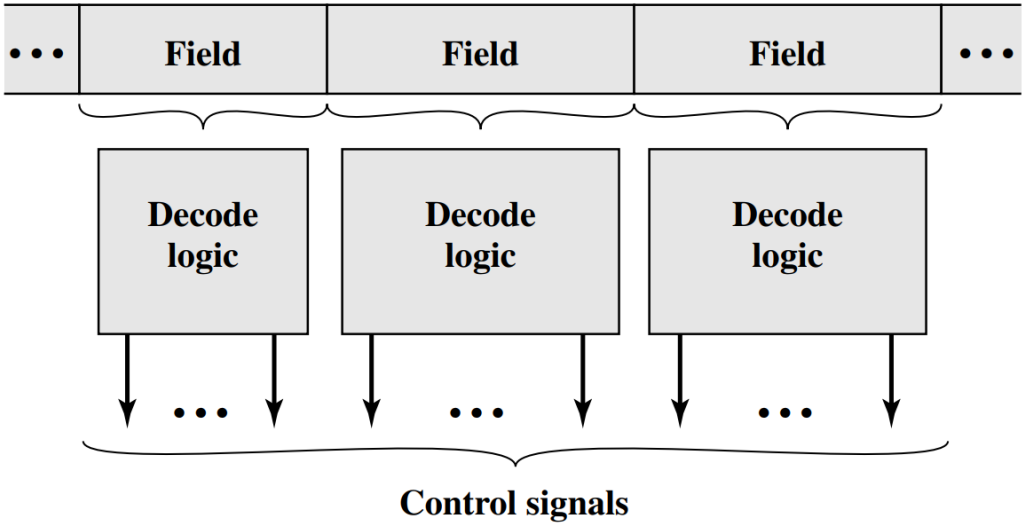
How to Encode -4

- In the design of microprogrammed controller, neither horizontal micro instruction format nor highly compact coding format is used 微程序控制器设计中，既不会使用水平微指令格式，也不会使用高度紧凑的编码格式
  - Use some encoding to reduce the width of control memory 使用某种程度的编码，减少控制存储器的宽度
  - Simplify microprogramming tasks 简化微程序设计任务
- Compromises 折中方案
  - More bits than necessary used 比需要使用的位数更多
  - Some combinations that are physically allowable are not possible to encode 某些物理上允许的组合，实际上不可能编码

Microinstruction encoding 微指令编码

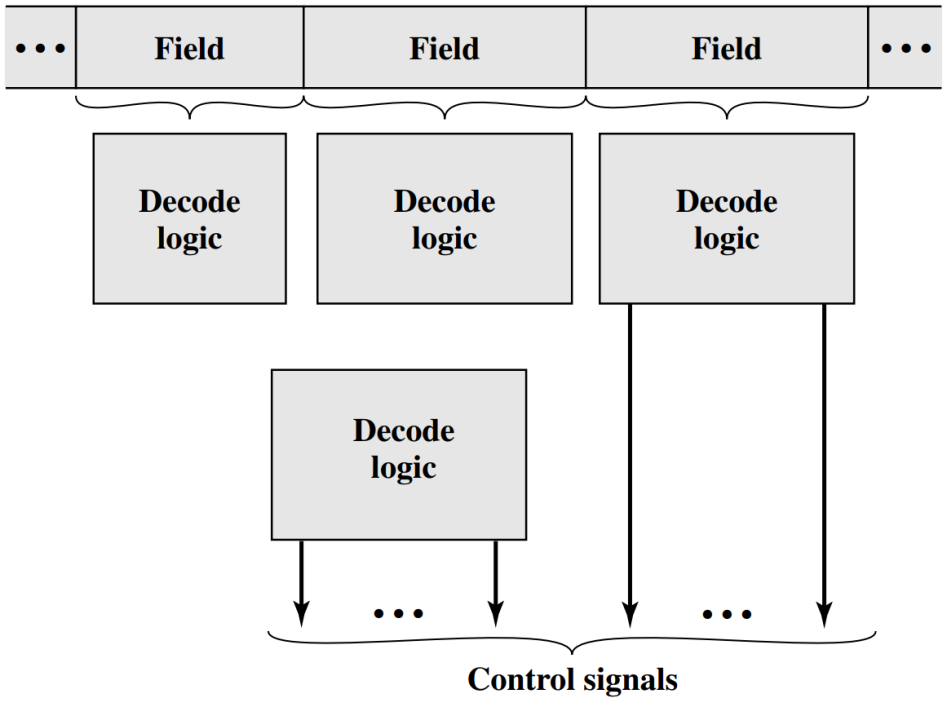
- Microinstruction organized as set of fields 微指令由一组字段组成
  - Each field contains code，activates one or more control signals 每个字段包含代码，激活一个或多个控制信号
- Organize format into independent fields 字段之间相互独立
  - Field depicts set of actions 字段描述一组动作
  - Actions from different fields can occur simultaneously 来自不同字段的操作可以同时发生
- Alternative actions that can be specified by a field are mutually exclusive 字段指定的操作是互斥的
  - Only one action specified for field could occur at a time 一次只能是一个字段指定操作才能发生

Direct encoding 直接编码



- 控制字中的各个字段直接通过译码逻辑，生成控制信号的一部分

Indirect encoding 间接编码



- 控制字中的多个字段，经过自身的一次译码逻辑之后，需要经过联合译码，才能得到控制信号
- 一个字段可能是另一个字段的解释

Summary 小结

- A microprogrammed control unit is a relatively simple logic circuit that is capable of 微程序控制器是一个相对简单的逻辑电路，能够
  - Sequencing through microinstructions 微指令定序
  - Generating control signals to execute each microinstruction 生成控制信号以执行每个微指令

Key Terms

Control Bus	Control signal	Hardwired implementation
Control Path	Control unit	microoperations

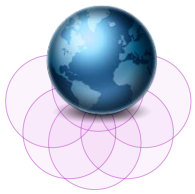
Control memory	Horizontal microinstruction	Microinstructions
Control word	Microinstruction encoding	microprogram
Fireware	Microinstruction execution	Microprogrammed control unit
Hard microprogramming	Microinstruction sequencing	Vertical microinstruction

Summary and Question

- 小结
  - 对控制器的工作原理进行了分析
  - 硬布线控制
  - 微程序控制
- 问题
  - 什么是微操作？
  - 硬布线方法的设计视角是什么？
  - 微程序的设计视角是什么？

Assignment

- Review
  - 15.2，15.5，15.8
  - 16.1，16.3，16.5，16.6
- Problem
  - 16.4



谢谢大家！