Windows Terminal + Windows Subsystem for Linux (WSL 2)

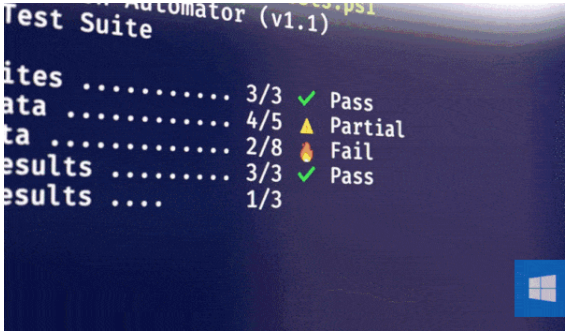
Windows的开源命令行工具

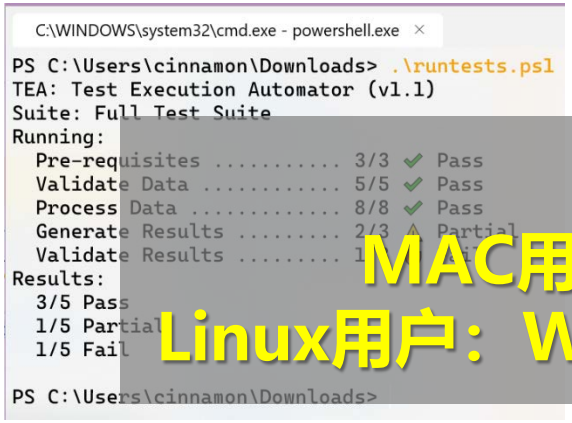
- 支持PowerShell，Cmd，WSL（Windows的Linux子系统）和SSH等命令行程序

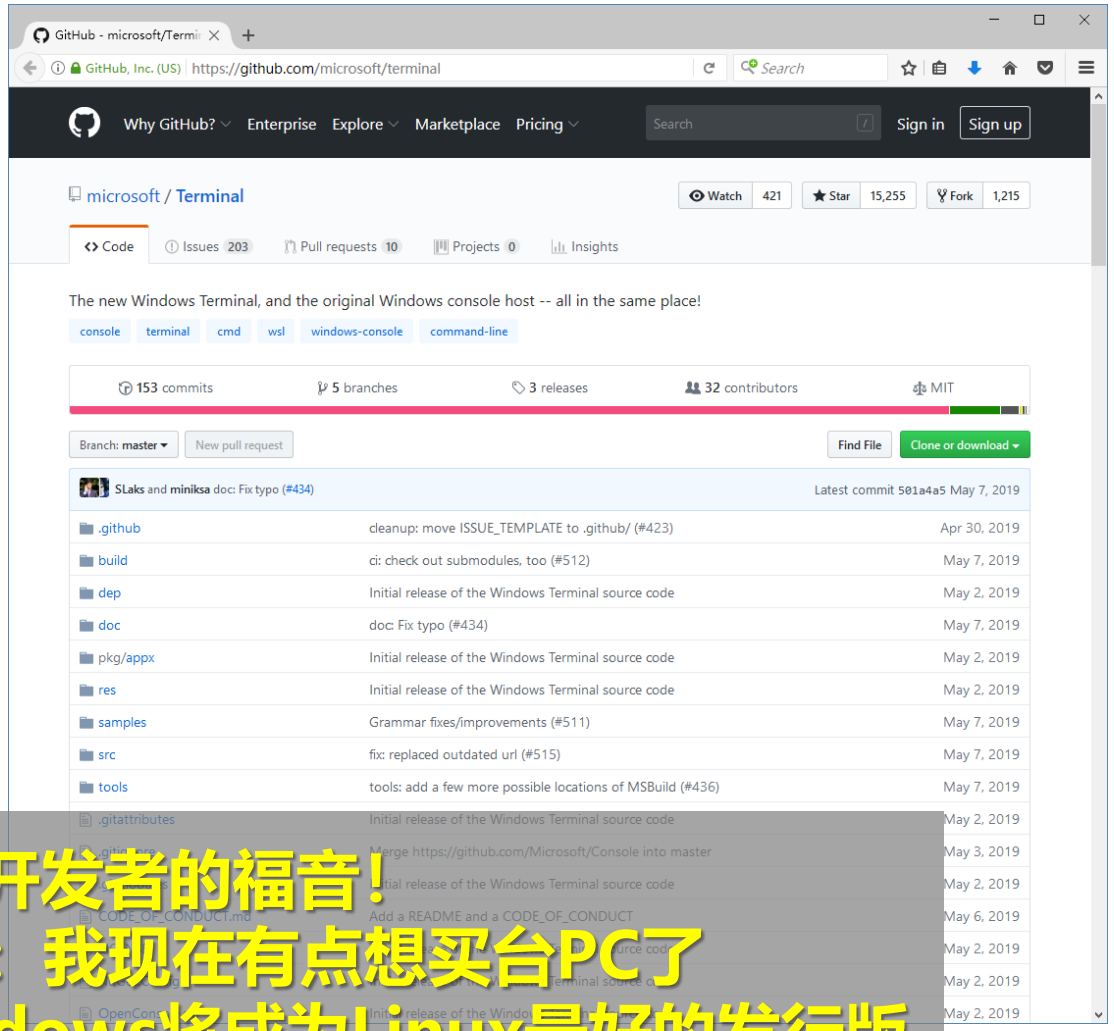
WSL2：Windows中完整真实的Linux内核

- 可运行ELF64 Linux binaries

** released at Microsoft Build 2019 developers conference on May 6, 2019*







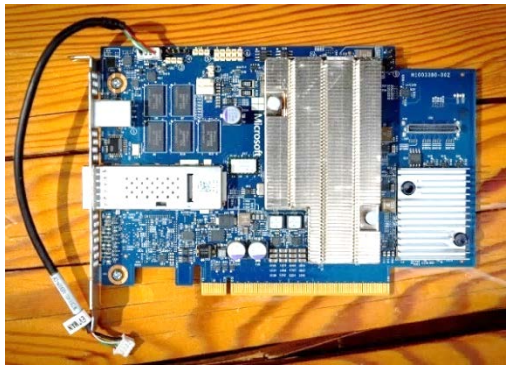
开发者的福音！

MAC用户：我现在有点想买台PC了

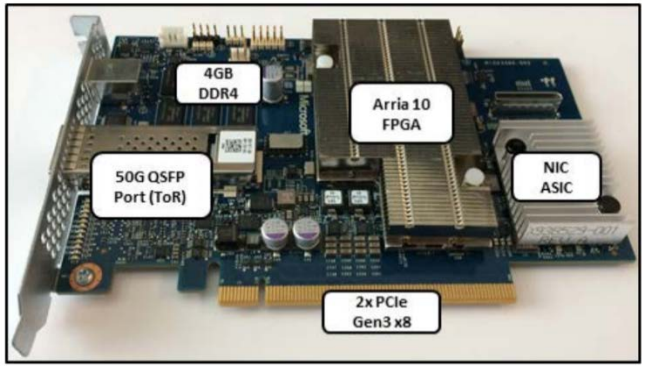
Linux用户：Windows将成为Linux最好的发行版

Microsoft Project Brainwave

- **Hardware architecture designed to accelerate real-time AI calculations**
  - Preview at Microsoft's Build 2018 developers conference (May 7, 2018)
  - Unveil at Hot Chips 2017 (Aug. 22, 2017)
  - Based on successful Project Catapult (back to 2010)



- **Three main layers of Project Brainwave system**
  - A high-performance, distributed system architecture
  - A hardware DNN engine synthesized onto FPGAs
  - A compiler and runtime for low-friction deployment of trained models



- **Deployed on Intel FPGA, to make real-time AI calculations at competitive cost and with the industry's lowest latency, or lag time**
  - Object classification using ResNet50 in less than 1.8 millisecond per image

复习（第九章）：

1. 一条指令四个工作周期内输出的操作命令
2. 处理器控制单元对外的输入及输出接口信号
3. 控制信号在完成一条指令的过程中所起的作用
   - 不采用 CPU 内部总线的方式
   - 采用 CPU 内部总线方式
4. 控制单元的多级时序系统
5. 指令周期、机器周期、节拍(状态)和时钟周期
6. 机器速度不仅与主频有关，还与机器周期中所含时钟周期（主频的倒数）数 以及 指令周期中所含的机器周期数有关
7. 产生不同微操作命令序列所用的时序控制方式
   - 同步控制：同步定长的机器周期、同步非定长的机器周期、中央和局部相结合
   - 异步控制
   - 同步与异步联合
   - 手工控制

B0911006Y-01　2018-2019学年春季学期

计算机组成原理

第 19 讲　控制单元的设计 I

如何通过组合逻辑方式与微程序方式设计控制单元?

主讲教师：　张　科

2019年5月8日

中国科学院大学 University of Chinese Academy of Sciences
中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY,CAS

第10章　控制单元的设计

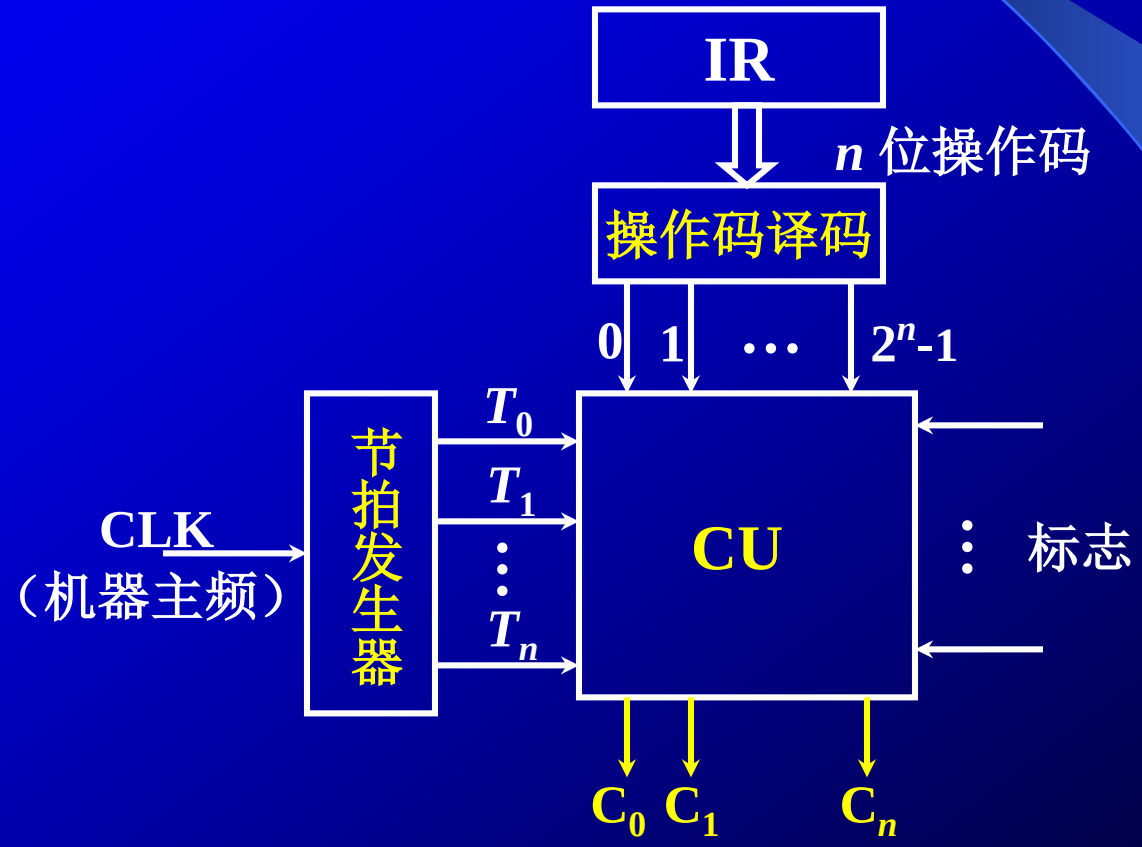
10.1　组合逻辑设计

10.2　微程序设计

10.1 组合逻辑设计

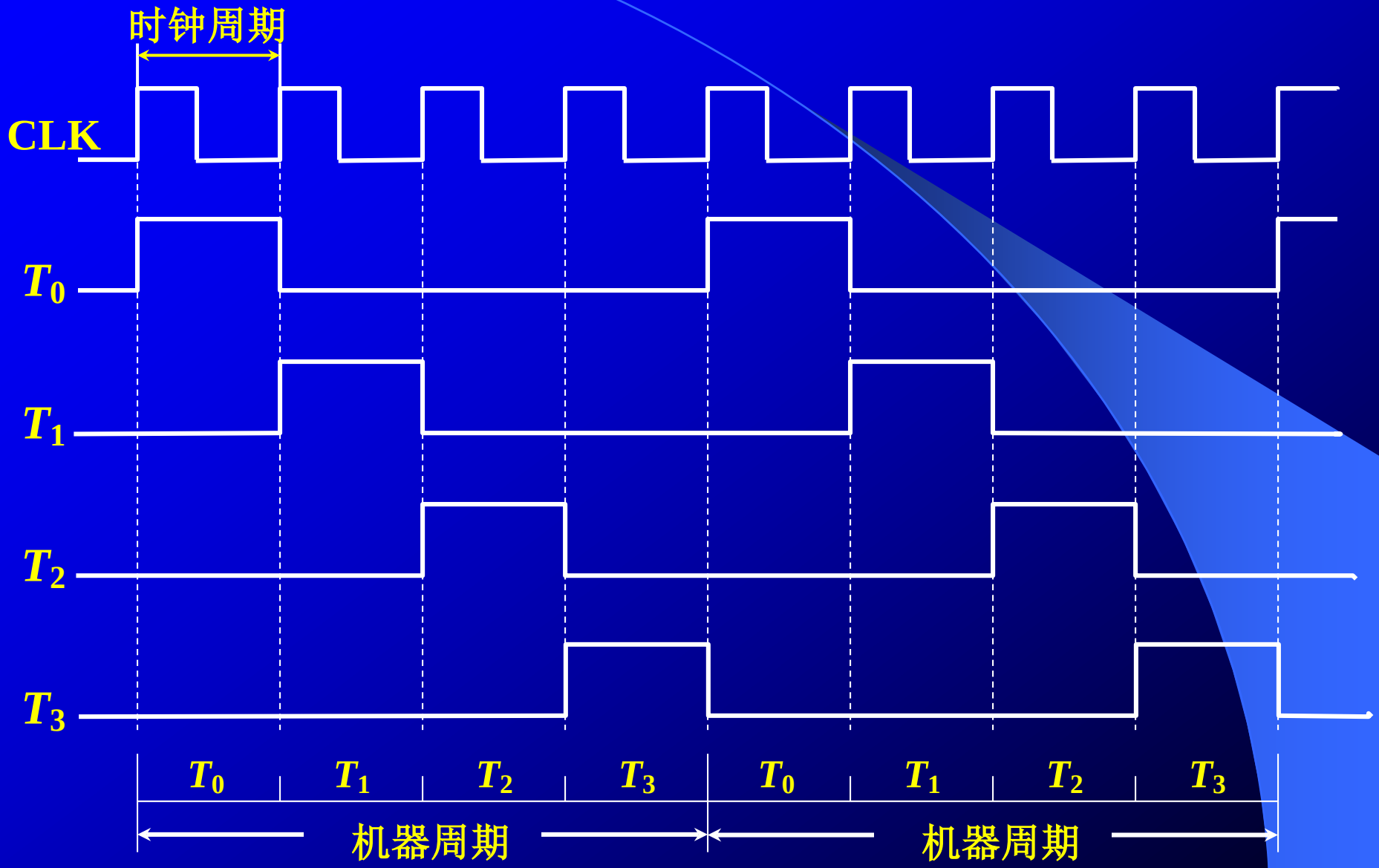
一、组合逻辑控制单元框图

1. CU 外特性



2. 节拍信号

10.1



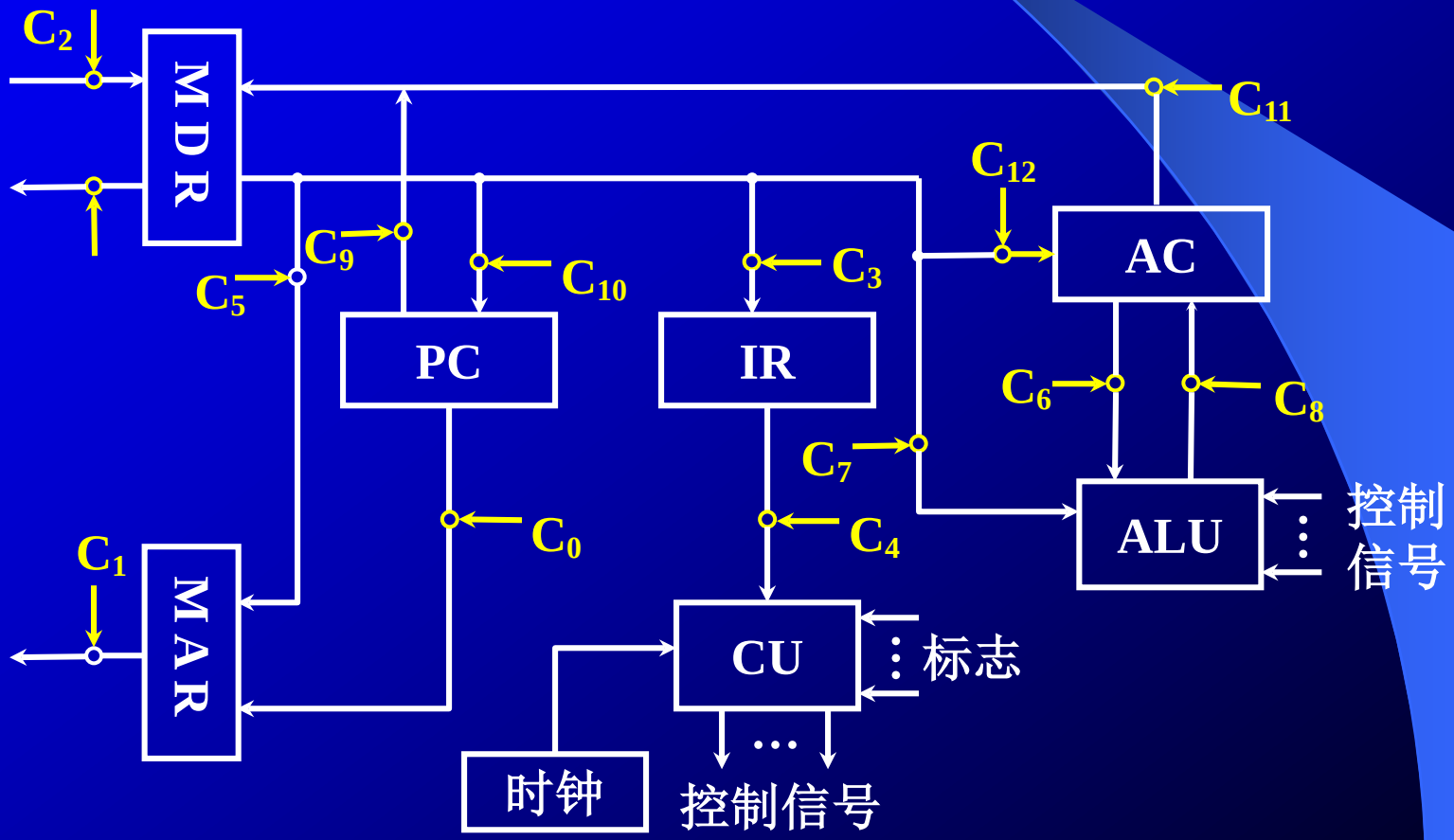
二、微操作的节拍安排

10.1

采用 同步控制方式

一个 机器周期 内有 3个节拍（时钟周期），定长

CPU 内部结构采用非总线方式



1. 安排微操作时序的原则

10.1

原则一　微操作的 先后顺序不得 随意 更改

原则二　被控对象不同 的微操作

尽量安排在 一个节拍 内完成

原则三　占用 时间较短 的微操作

尽量 安排在 一个节拍 内完成

并允许有先后顺序

2. 取指周期 微操作的 节拍安排

T_0 PC ⟶ MAR　　原则二

1 ⟶ R

T_1 M (MAR) ⟶ MDR　　原则二

(PC) + 1 ⟶ PC

T_2 MDR ⟶ IR　　原则三

OP (IR) ⟶ ID

3. 间址周期 微操作的 节拍安排

T_0 Ad (IR) ⟶ MAR

1 ⟶ R

T_1 M (MAR) ⟶ MDR

T_2 MDR ⟶ Ad (IR)

4. 执行周期 微操作的 节拍安排

10.1

① CLA T_0

T_1

T_2 $0 \longrightarrow AC$

② COM T_0

T_1

T_2 $\overline{AC} \longrightarrow AC$

③ SHR T_0

T_1

T_2 $L(AC) \longrightarrow R(AC)$

$AC_0 \longrightarrow AC_0$

④ CSL	T_0		
	T_1		
	T_2	R (AC) ⟶ L (AC)	$AC_0 \longrightarrow AC_n$
⑤ STP	T_0		
	T_1		
	T_2	0 ⟶ G	
⑥ ADD X	T_0	Ad (IR) ⟶ MAR	1 ⟶ R
	T_1	M (MAR) ⟶ MDR	
	T_2	(AC) + (MDR) ⟶ AC	
⑦ STA X	T_0	Ad (IR) ⟶ MAR	1 ⟶ W
	T_1	AC ⟶ MDR	
	T_2	MDR ⟶ M (MAR)	

⑧ LDA X

T_0 Ad (IR) ⟶ MAR　1 ⟶ R

T_1 M (MAR) ⟶ MDR

T_2 MDR ⟶ AC

⑨ JMP X

T_0

T_1

T_2 Ad (IR) ⟶ PC

⑩ BAN X

T_0

T_1

T_2 $A_0 \cdot Ad(IR) + \overline{A}_0 \cdot PC \longrightarrow PC$

执行周期的最后时刻，CPU若检测到中断源有请求，则在允许中断的条件下，CPU进入中断周期

5. 中断周期 微操作的 节拍安排

T_0　0 ⟶ MAR　　1 ⟶ W　　0 ⟶ EINT（置“0”）
硬件关中断

T_1　PC ⟶ MDR

T_2　MDR ⟶ M (MAR)　　向量地址 ⟶ PC

上述微操作中断隐指令完成（参考8.4.4节）

在机器指令系统中没有的指令
CPU 在中断周期内由硬件自动完成

三、组合逻辑设计步骤

10.1

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
FE 取指	T_0		PC ⟶ MAR						
			1 ⟶ R						
	T_1		M(MAR) ⟶ MDR						
			(PC) +1 ⟶ PC						
	T_2		MDR ⟶ IR						
			OP(IR) ⟶ ID						
		I	1 ⟶ IND						
		$\bar{I}$	1 ⟶ EX						

间址特征

FE/IND/EX为CPU工作周期标志、状态寄存器，P343图8.9

三、组合逻辑设计步骤

10.1

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
IND 间址	T_0		Ad (IR) ⟶ MAR						
			1 ⟶ R						
	T_1		M(MAR) ⟶ MDR						
	T_2		MDR ⟶ Ad (IR)						
		$\overline{IND}$	1 ⟶ EX						

间址周期标志

IND=0为一次间接寻址；IND=1为多次间接寻址

三、组合逻辑设计步骤

10.1

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
EX 执行	T_0		Ad (IR)⟶ MAR						
			1 ⟶ R						
			1 ⟶ W						
	T_1		M(MAR)⟶ MDR						
			AC ⟶ MDR						
	T_2		(AC)+(MDR)→AC						
			MDR ⟶ M(MAR)						
			MDR⟶ AC						
			0⟶ AC						

三、组合逻辑设计步骤

10.1

1. 列出操作时间表（若某指令存在所列微操作则填1）

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
FE 取指	T_0		PC ⟶ MAR	1	1	1	1	1	1
			1 ⟶ R	1	1	1	1	1	1
	T_1		M(MAR) ⟶ MDR	1	1	1	1	1	1
			(PC) +1 ⟶ PC	1	1	1	1	1	1
	T_2		MDR ⟶ IR	1	1	1	1	1	1
			OP(IR) ⟶ ID	1	1	1	1	1	1
		I	1 ⟶ IND			1	1	1	1
		$\bar{I}$	1 ⟶ EX	1	1	1	1	1	1

三、组合逻辑设计步骤

10.1

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
IND 间址	T_0		Ad (IR)⟶ MAR			1	1	1	1
			1⟶ R			1	1	1	1
	T_1		M(MAR)⟶ MDR			1	1	1	1
	T_2		MDR⟶ Ad (IR)			1	1	1	1
		$\overline{\text{IND}}$	1⟶ EX			1	1	1	1

三、组合逻辑设计步骤

10.1

1. 列出操作时间表

工作周期标记	节拍	状态条件	微操作命令信号	CLA	COM	ADD	STA	LDA	JMP
EX 执行	T_0		Ad (IR)→ MAR			1	1	1	
			1→ R			1		1	
			1 → W				1		
	T_1		M(MAR)→ MDR			1		1	
			AC → MDR				1		
	T_2		(AC)+(MDR)→AC			1			
			MDR → M(MAR)				1		
			MDR→ AC					1	
			0→ AC	1					

2. 写出微操作命令的最简表达式 10.1

M (MAR) ⟶ MDR

= FE · T_1 + IND · T_1 (ADD + STA + LDA + JMP + BAN)

+ EX · T_1 (ADD +LDA)

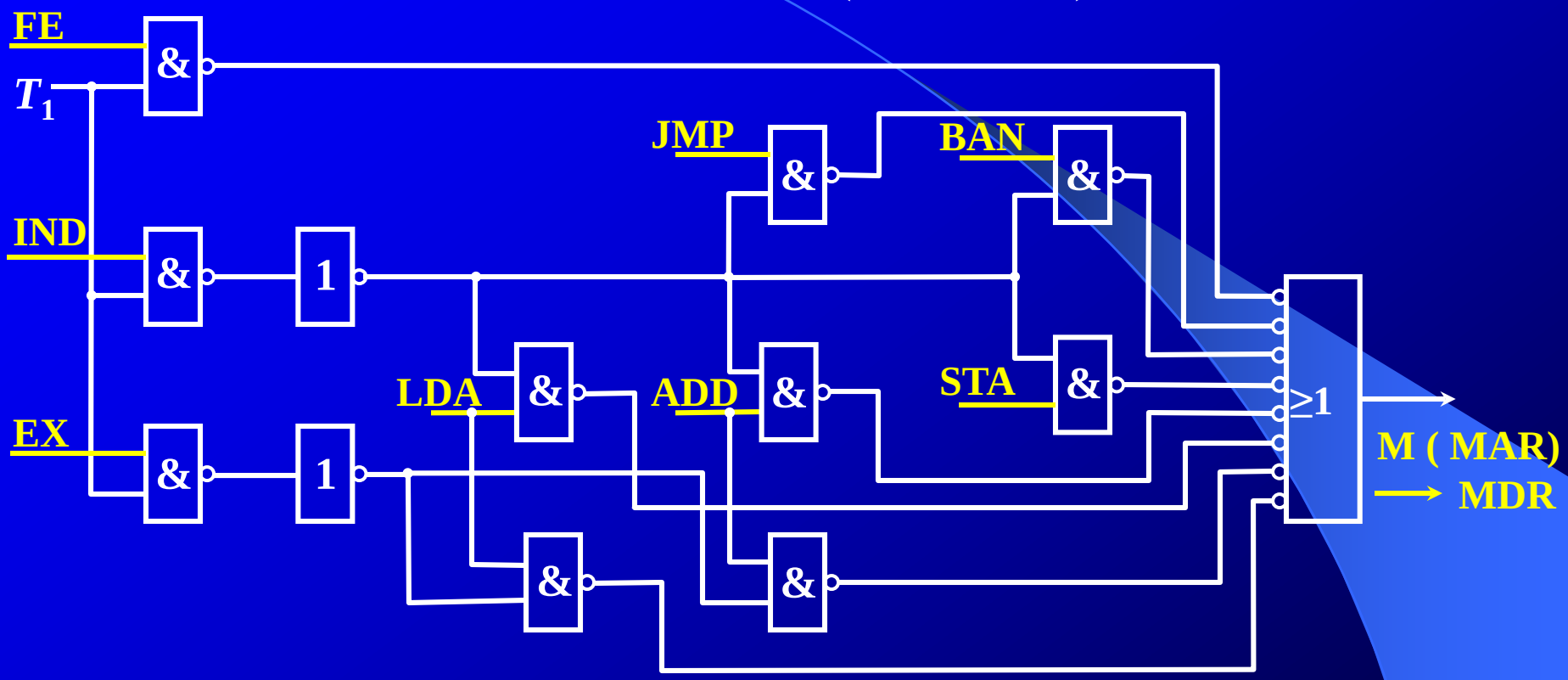
= T_1{ FE + IND (ADD + STA + LDA + JMP + BAN)

+ EX (ADD +LDA) }

3. 画出逻辑图

$$M(MAR) \longrightarrow MDR$$

$$= T_1\{FE + IND(ADD + STA + LDA + JMP + BAN) + EX(ADD + LDA)\}$$



特点

- 思路清晰，简单明了
- 速度快 （RISC）
- 庞杂，调试困难，修改困难

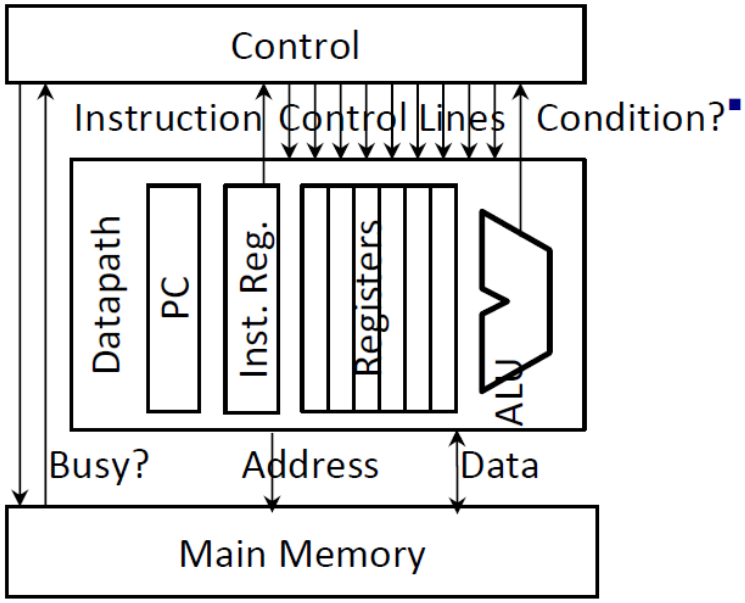
第10章　控制单元的设计

10.1　组合逻辑设计

10.2　微程序设计

Control versus Datapath

- Processor designs split between *datapath*, where numbers are stored and arithmetic operations computed, and *control*, which sequences operations on datapath
- Biggest challenge for computer designers was getting control correct



- **Maurice Wilkes** invented the idea of *microprogramming* to design the control unit of a processor*

  - Logic expensive vs. ROM or RAM
  - ROM cheaper and faster than RAM
  - *Control design now programming*

* "Micro-programming and the design of the control circuits in an electronic digital computer," M. Wilkes, and J. Stringer. *Mathematical Proc. of the Cambridge Philosophical Society*, Vol. 49, 1953.

A New Golden Age for Computer Architecture: History, Challenges, and Opportunities. David Patterson, December 5, 2018

10.2 微程序设计

一、微程序设计思想（Microprogramming/Microcode）

1951年 由英国剑桥大学教授 Sir Maurice Vincent Wilkes提出
1958年 EDSAC2中使用microprogrammed/microcode control unit

Electronic Delay Storage Automatic Calculator

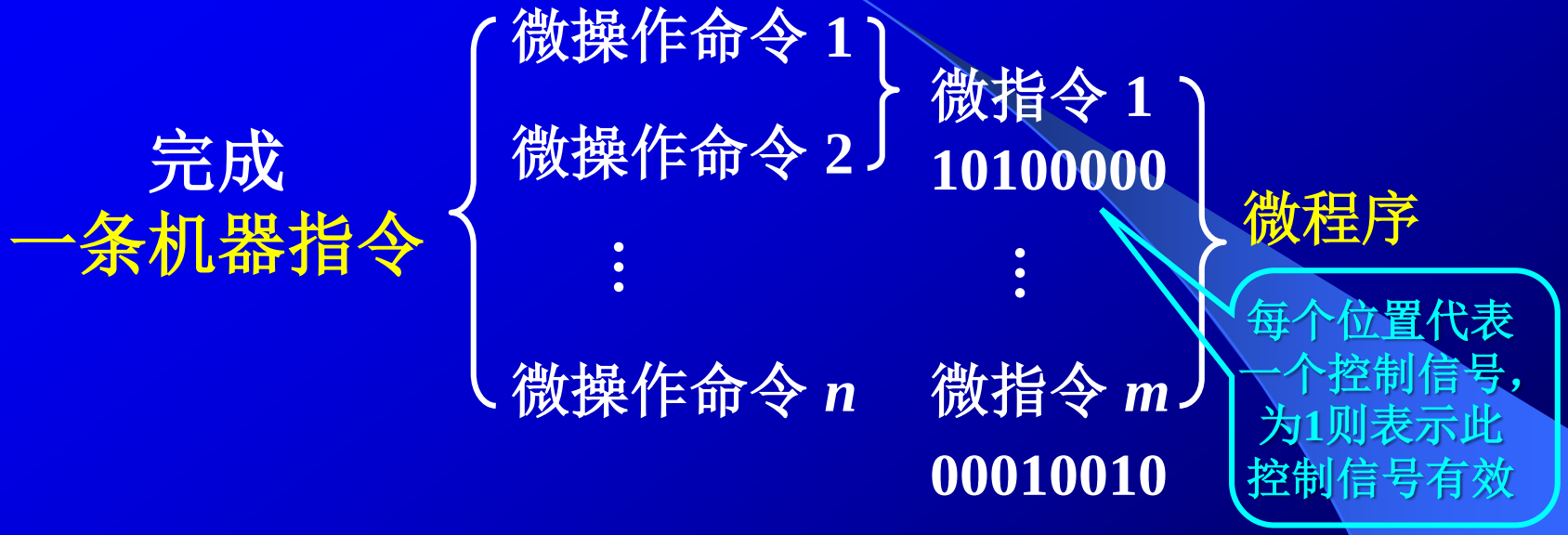
(1913-2010)

- 第二届图灵奖获得者 (1967)
- the builder and designer of the **EDSAC** (1949/05/06), the first complete and fully operational regular electronic digital computer with an internally **stored program**.
- Program libraries (1951)

- 1st paper on cache memories (1965)
- Time-sharing Computer Systems & client-server architecture computing (1975)

First Draft of a Report on the EDVAC "Electronic Discrete Variable Automatic Computer" (1945)
Operation of EDVAC (1951)

10.2 微程序设计



一条机器指令对应一个微程序　　存入 控制存储器（使用 ROM实现）

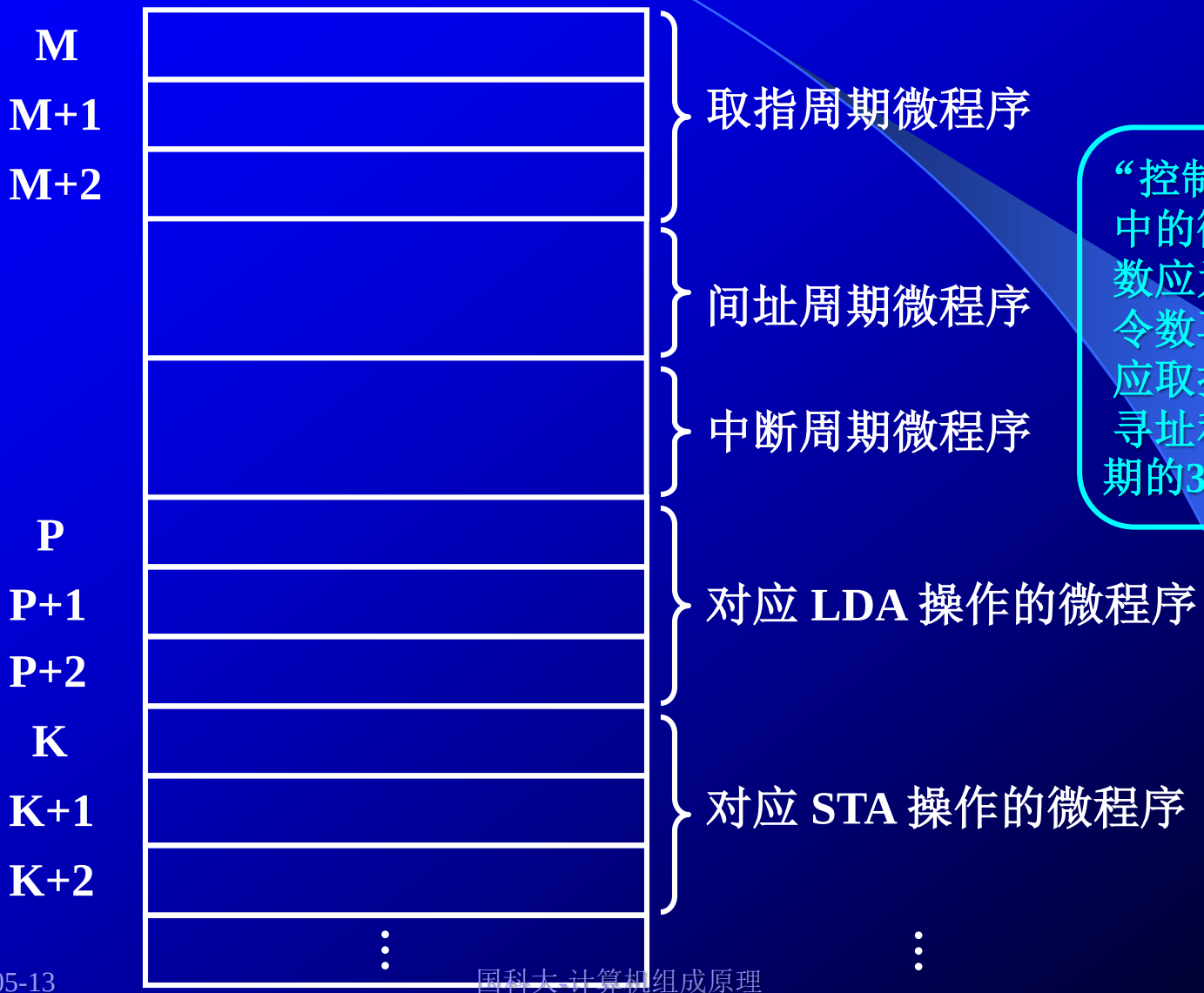
“存储逻辑”　　1964年 IBM System/360

存储控制

二、微程序控制单元框图及工作原理

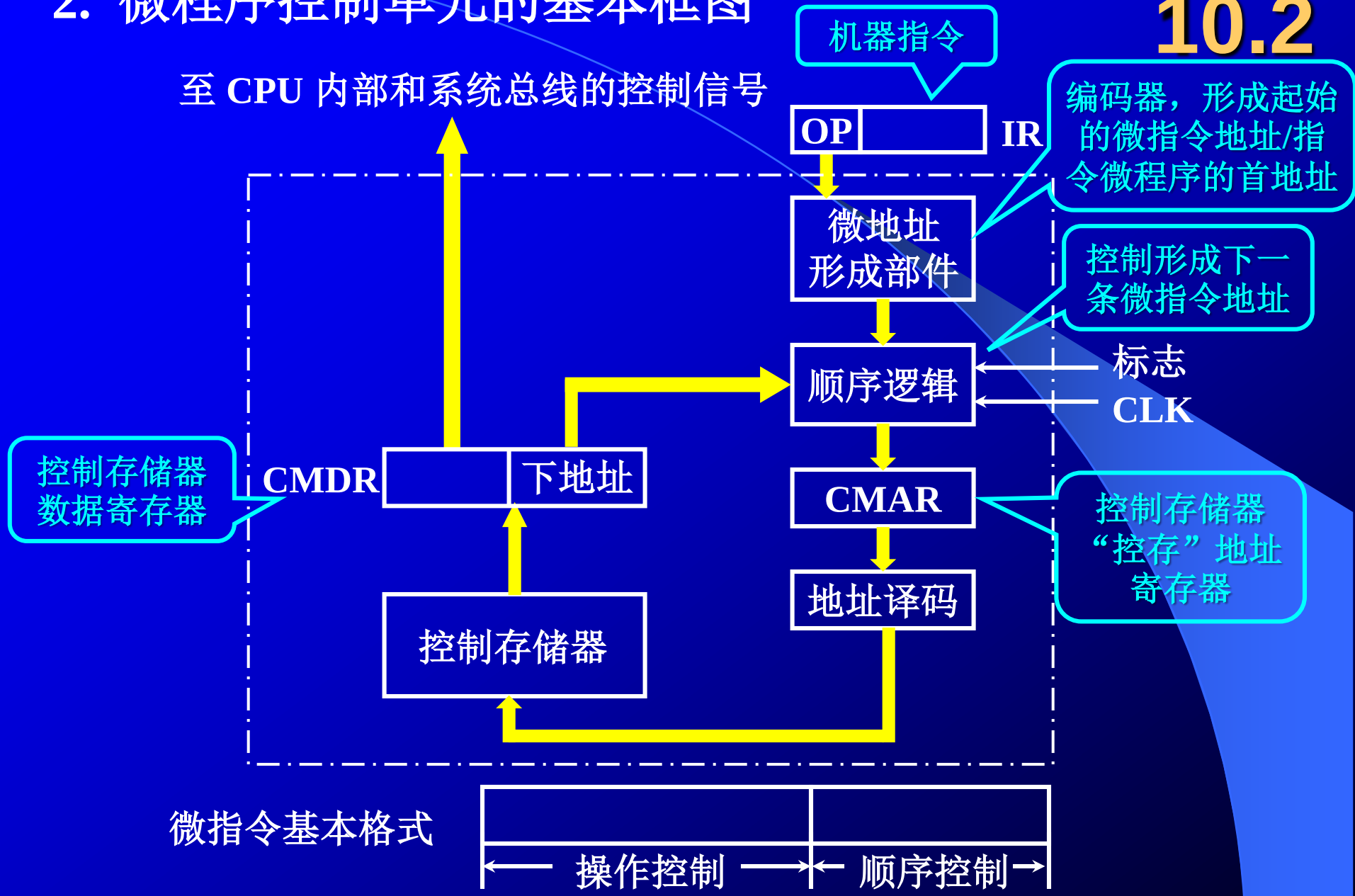
10.2

1. 控制存储器中机器指令对应的微程序



“控制存储器”中的微程序个数应为机器指令数再加上对应取指、间接寻址和中断周期的3个微程序

2. 微程序控制单元的基本框图



二、微程序控制单元框图及工作原理

10.2

3. 工作原理


主存
用户程序
LDA X
ADD Y
STA Z
STP


控存
M
M+1
M+2
P
P+1
P+2
Q
Q+1
Q+2
K
K+1
K+2
M+1
M+2
×××
P+1
P+2
M
Q+1
Q+2
M
K+1
K+2
M
取指周期微程序
对应 LDA 操作的微程序
对应 ADD 操作的微程序
对应 STA 操作的微程序

3. 工作原理

(1) 取指阶段　　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址　M + 1

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

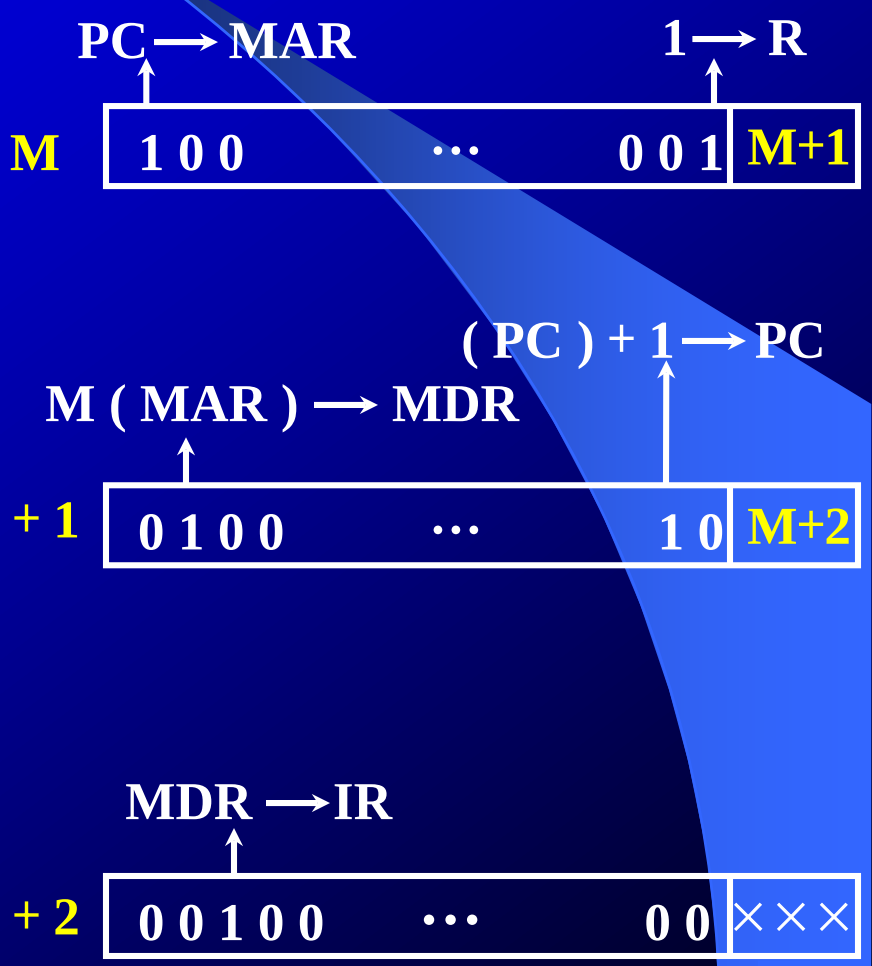
由 CMDR 发命令

形成下条微指令地址　M + 2

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令



(2) 执行阶段　执行 LDA 微程序

10.2

LDA指令微程序首地址

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR　(P ⟶ CMAR)

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 P + 1

Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 P + 2

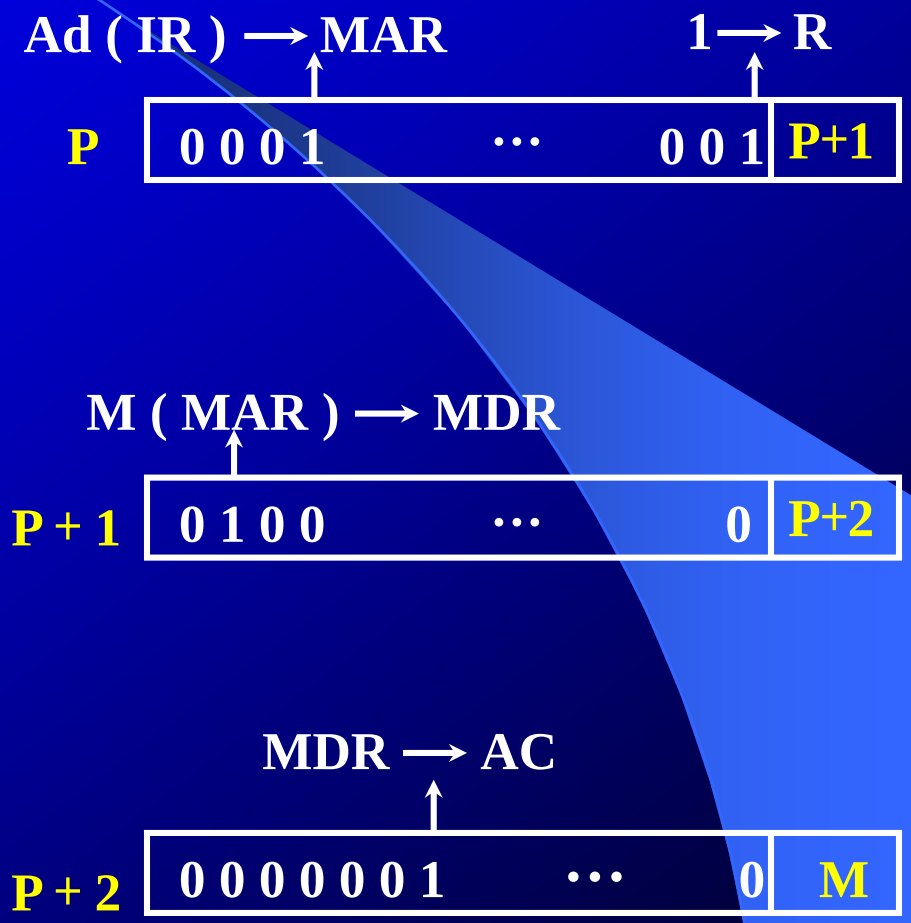
Ad (CMDR) ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

形成下条微指令地址 M

Ad (CMDR) ⟶ CMAR　(M ⟶ CMAR)



取指周期微程序首地址

(3) 取指阶段　执行取指微程序

M ⟶ CMAR

CM (CMAR) ⟶ CMDR

由 CMDR 发命令

⋮

全部微指令存在 CM 中，程序执行过程中 只需读出
因此，可以采用ROM只读存储器实现控存CM

关键

- 微指令的 操作控制字段如何形成微操作命令 (即如何编码)
- 微指令的 后续地址如何形成

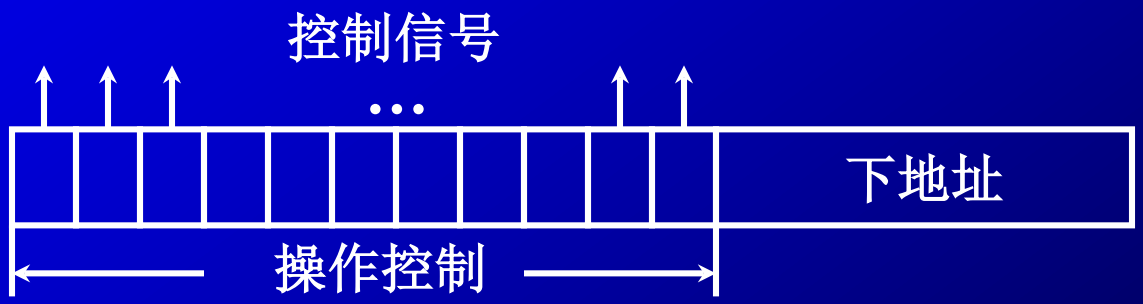
三、微指令的编码方式（控制方式）

10.2

1. 直接编码（直接控制、不译法）方式

在微指令的操作控制字段中，

每一位代表一个微操作命令

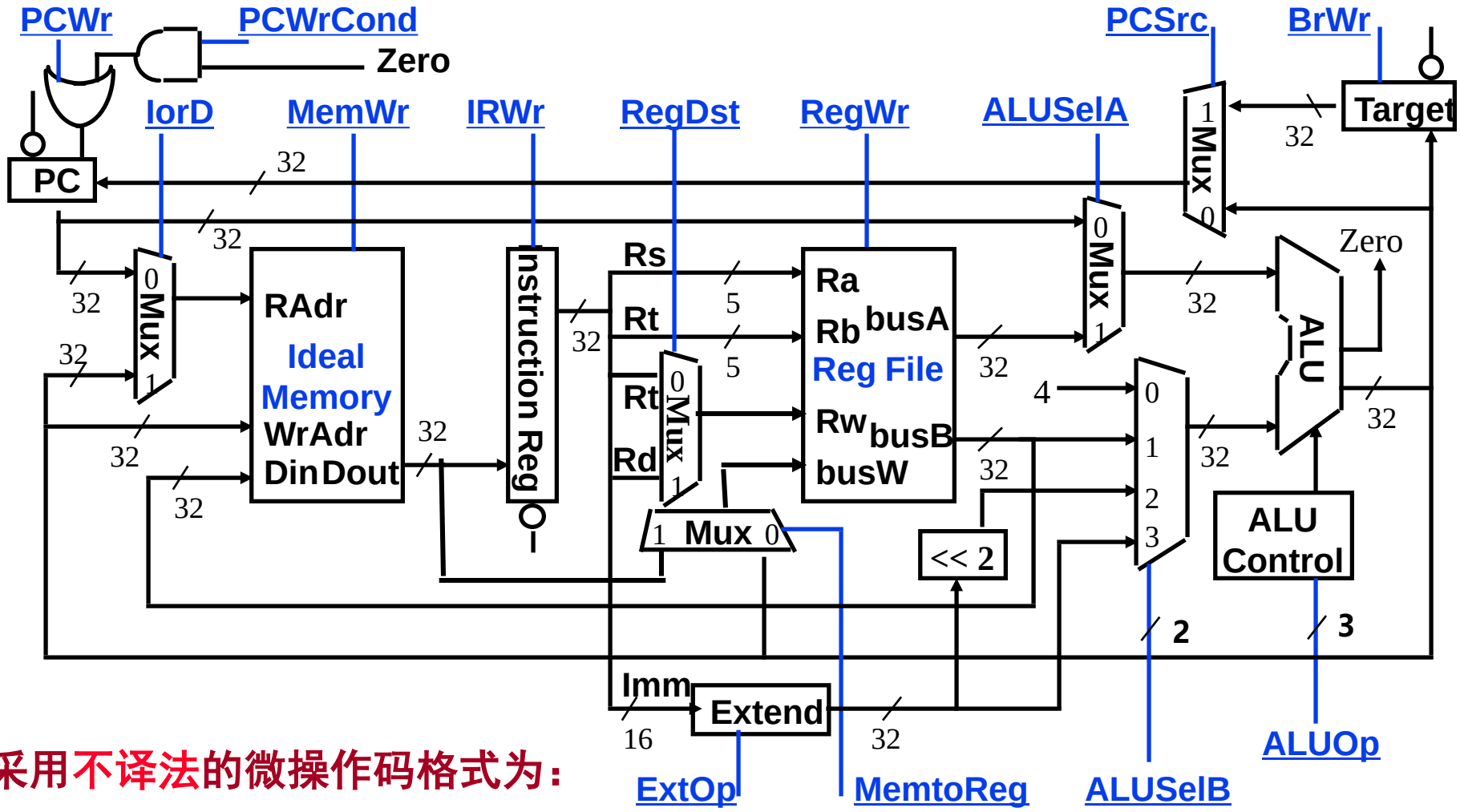


Wilkes微程序控制器是最早提出的，采用的就是直接控制法

某位为 “1” 表示该控制信号有效

速度最快，但是存储容量消耗大

多周期数据通路对应的微操作码



采用不译法的微操作码格式为：

PCWr	IorD	MemWr	……	PCSrc	BrWr	……

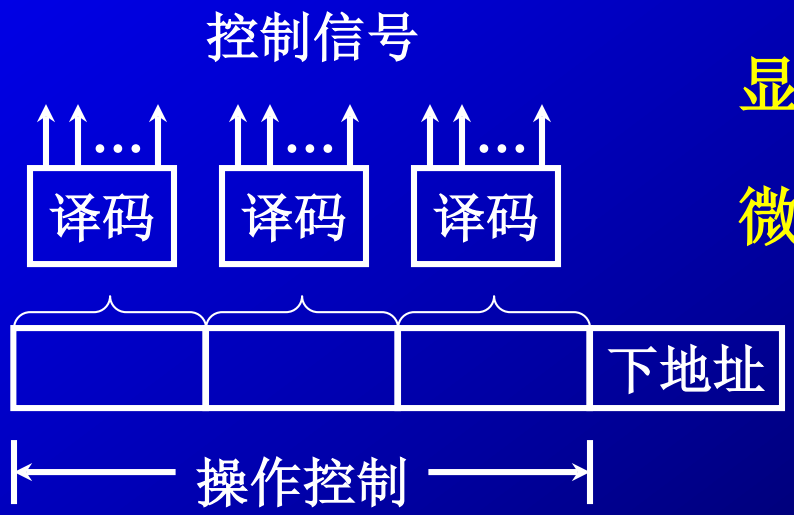
控制字 (微指令) 长度等于控制信号（微命令）总位数

该图对应控制器的微指令长度为多少位？ 17位

2. 字段直接编码方式

将微指令的控制字段分成若干 “段”，

每段经译码后发出控制信号



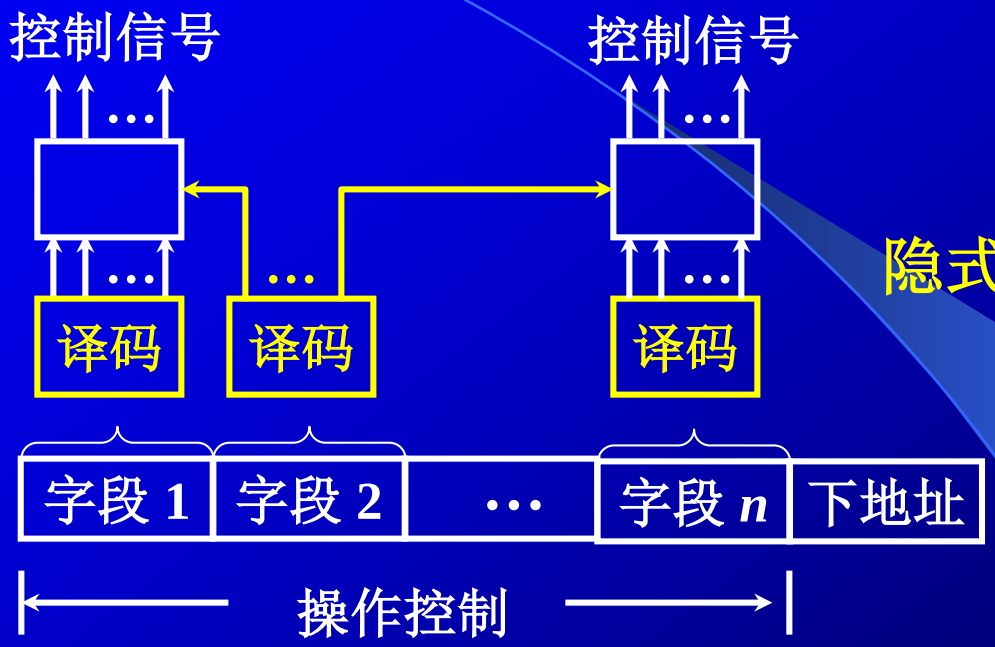
显式编码

微程序执行速度较慢

每个字段译码出的微操作命令是 互斥 的，字段之间为并行发出

缩短 了微指令 字长，增加 了译码 时间

3. 字段间接编码方式



4. 混合编码

直接编码和字段编码（直接和间接）混合使用

5. 其他（如常数字段）

四、微指令序列地址的形成

1. 微指令的 下地址字段 指出

2. 根据机器指令的 操作码 形成

3. 增量计数器

$$(CMAR) + 1 \longrightarrow CMAR$$

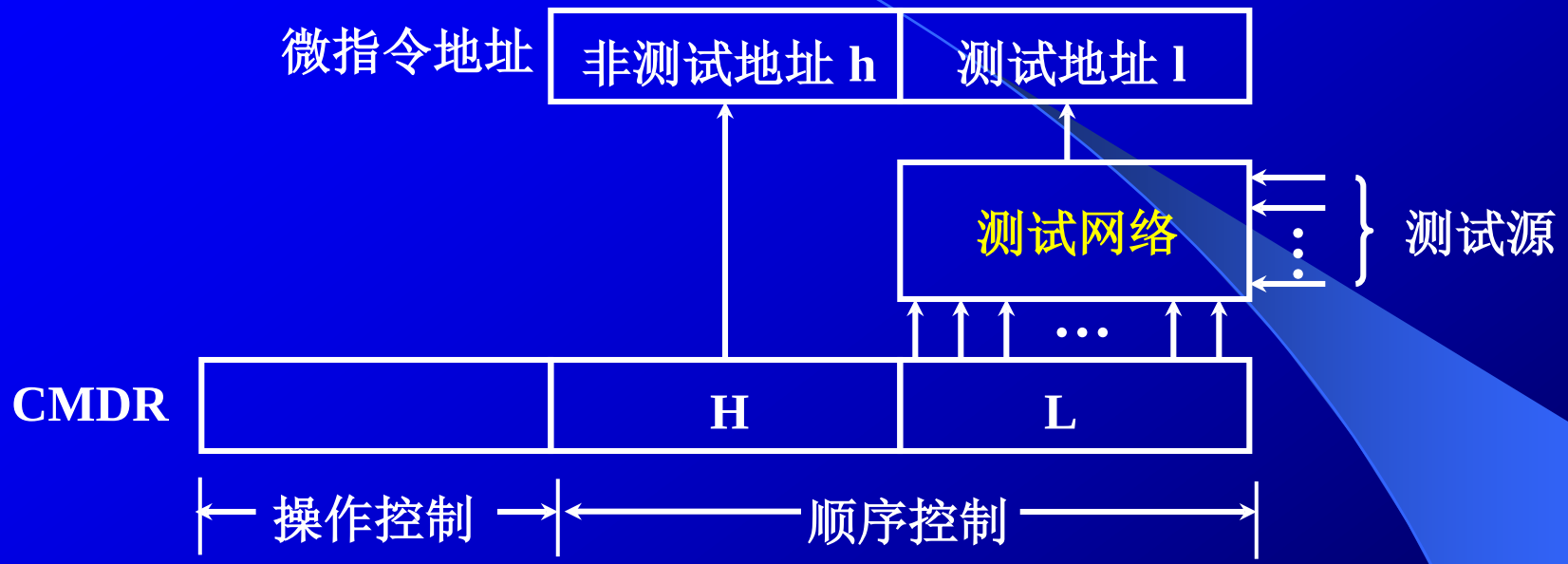
4. 分支转移

操作控制字段	转移方式	转移地址

转移方式　　指明判别条件

转移地址　　指明转移成功后的去向

5. 通过测试网络



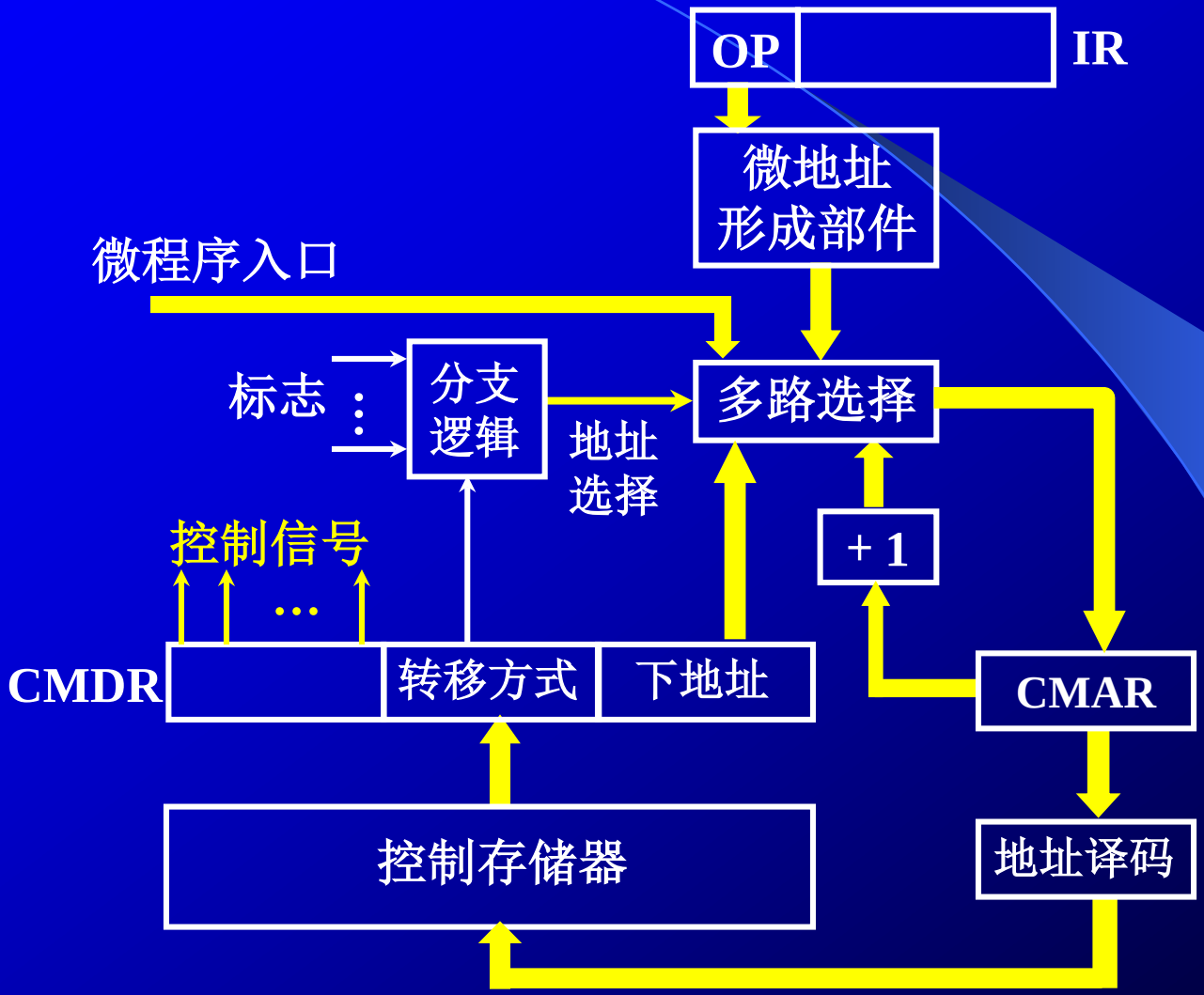
6. 由硬件产生微程序入口地址

第一条微指令地址　　由专门 硬件 产生

中断周期　　由 硬件 产生 中断周期微程序首地址

7. 后续微指令地址形成方式原理图

10.2



作业

- 习题：10.2，10.7，10.8
- 本次作业提交截止时间：
  - 请于5月13日上课前提交

蚂蚁金服的BASIC（基石）

金融科技的基石

- **Blockchain （区块链）**
- **Artificial Intelligence（人工智能）**
- **Security（安全）**
- **IoT（物联网）**
- **Cloud computing（云计算）**

蚂蚁金服十五年技术演进之路，2019/5/6
QCon北京2019，胡喜 (蚂蚁金服副CTO)

蚂蚁金服的basic（基础）

“有一个事情对我有很大触动。

我听别人说：Google 为了做好 TPU 的性能优化，从全公司能够紧急调集几十个做编译器的人才，而且这个还不是全部编译器人才，听说只是挑选了一些。今天，国内到底哪家公司能够拿出几十个这样的人才？我相信是很少的。

而今天中国整个软件业的发展，一定是非常需要计算机体系架构方面的人才，尤其是编译器，操作系统，硬件方面的人才，软件领域每一次重大变化都会带来一次重构和抽象，这个都会需要具备这样素质的人才。

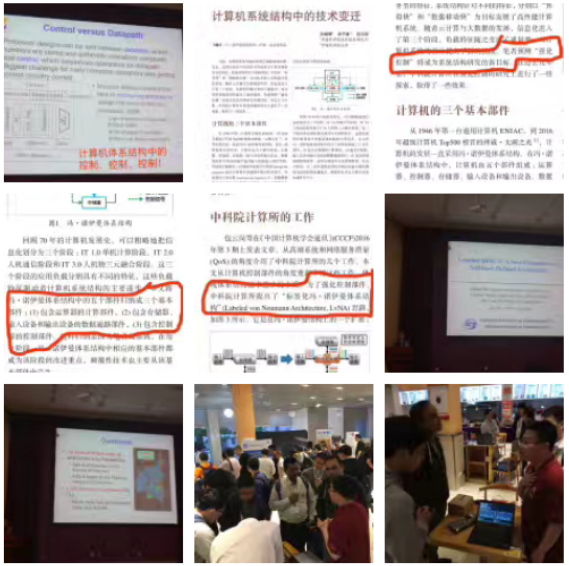
所以我觉得人才是非常非常关键的，只有找到这些人才，才能把这些事情好，才能做一些大的变化，才能对未来的不确定性做一些准备。”

发现　　朋友圈

包云岗

David Patterson的这张胶片必须单独拎出来——人们长期忽视了对计算机体系结构中控制部分的研究，但实际上控制也是冯诺伊曼体系结构中核心之一！其实，1958年Maurice Wilkes(第二届图灵奖得主)就提出了microprogramming思想来设计控制部件。看到这页PPT，我像被闪电击中了，好像一下子领悟到60年前计算机先驱的思想多么深刻啊——我们近年来的工作其实本质上就是针对共享资源的microprogramming啊！

收起



2017年5月9日

复习：

1. Microprogramming微程序控制器的基本思想：
   - 基于“存储程序”的思路，仿照程序设计的方法，编制每个指令对应的微程序→“存储逻辑、存储控制”
   - 每个微程序由若干条微指令构成，各微指令包含若干条微操作命令（一条微指令相当于一个状态/时钟/节拍，一个微操作就是状态中的控制信号）
   - 所有指令对应的微程序放在只读存储器（ROM）中，执行某条指令时，依次取出对应微程序中的各条微指令。对微指令译码产生对应的微命令，这个微命令就是控制信号。这个ROM称为控制存储器，简称控存。
2. 最初把固化在ROM中的微程序或程序称为固件（Firmware），表示用软件实现的硬部件；现在对固件通俗的理解是在ROM中“固化的软件”。
3. 微指令设计的两个关键点：操作控制字段如何编码、后续地址如何产生

B0911006Y-01　2018-2019学年春季学期

计算机组成原理

第20讲(上)　控制单元的设计II

如何设计微指令?

主讲教师:　张　科

2019年5月13日

中国科学院大学 University of Chinese Academy of Sciences
中科院计算所 INSTITUTE OF COMPUTING TECHNOLOGY,CAS

五、微指令格式

10.2

1. 水平型微指令 (Horizontal microcode)

一次能定义并执行多个并行操作 (as in Wilkes' original proposal)

如　直接编码、字段直接编码、字段间接编码、
直接和字段混合编码

2. 垂直型微指令 (Vertical microcode)

类似机器指令操作码 的方式

由微操作码字段规定微指令的功能

微操作编码方式称为：最小（最短、垂直）编码（译）法

2. 垂直型微指令示例

表 10.2　垂直型微指令示例

微操作码	地址码		其他		微指令类别及功能
0 1 2	3 ~ 7	8 ~ 12	13　15		
0 0 0	源寄存器	目的寄存器	其他控制		传送型微指令
0 0 1	ALU 左输入	ALU 右输入	ALU		运算控制型微指令 按 ALU 字段所规定的功能执行,其结果送暂存器
0 1 0	寄存器	移位次数	移位方式		移位控制型微指令 按移位方式对寄存器中的数据移位
0 1 1	寄存器	存储器	读写	其他	访存微指令 完成存储器和寄存器之间的传送
1 0 0	D			S	无条件转移微指令 D 为微指令的目的地址
1 0 1	D		测试条件		条件转移微指令 最低 4 位为测试条件
1 1 0 1 1 1					可定义 I/O 或其他操作 第 3 ~ 15 位可根据需要定义各种微命令

Vertical microcode: Microdata example

```
; multiply two numbers

      LF   5,X'08'    ; set loop counter to 8
      MT   2          ; move X to T
      LF   4,X'00'    ; clear ZU
ADD:  TZ   3,X'01'    ; if Y bit 0 set to 1
      A    4,T        ; add X to Z
      H    4,R        ; shift Z upper
      H    3,L,R      ; shift Z lower
      D    5,C        ; decrement loop counter
      TN   0,X'04'    ; if loop counter equal 0
      JP   ADD        ; jump to top of loop
```

Horizontal microcode: Nanodata example

```
SRDAI: "SHIFT RIGHT DOUBLE ARITHMETIC IMMEDIATE"

.... FETCH, KSHC=RIGHT+DOUBLE+ARITHMETIC+RIGHT CTL,  SH STATUS ENABLE
            KALC=PASS LEFT,                        ALU STATUS ENABLE
S...    A->FAIL,    A->FSID, B->KSHR, CLEAR CIN
.S..    A->FAOD, INCF->FSID, A->FSOD
..S. INCF->FSOD
...X GATE ALU, GATE SH, ALU TO COM
```

3. 水平型微指令的特点

(1) 水平型微指令比垂直型微指令 并行操作能力强，灵活性强

(2) 水平型微指令执行一条机器指令所要的微指令 数目少，速度快

(3) 水平型微指令 用较长的微指令结构换取较短的微程序结构

(4) 水平型微指令与机器指令 差别大

两种微指令格式的比较

- **水平型微指令**

  基本思想：相容微命令尽量多地安排在一条微指令中。

  优点：微程序短，并行性高，适合于较高速度的场合。

  缺点：微指令长，编码空间利用率较低，并且编制困难。

- **垂直型微指令**

  基本思想：一条微指令只控制一、二个微操作命令。

  优点：微指令短，编码效率高，格式与机器指令类似，故编制容易。

  缺点：微程序长，一条微指令只能控制一、二个微操作命令，无并行，速度慢。

- **水平型微指令面向控制逻辑描述**
- **垂直型微指令面向控制算法描述**

六、静态微程序设计和动态微程序设计

10.2

静态　微程序无须改变，采用 **ROM**

动态　通过 改变微指令 和 微程序 改变机器指令，有利于仿真，采用 **EPROM**

可擦可编程只读存储器（Erasable Programmable Read-Only-Memory）意味着微程序可以后续被更新

七、毫微程序设计

1. 毫微程序设计的基本概念

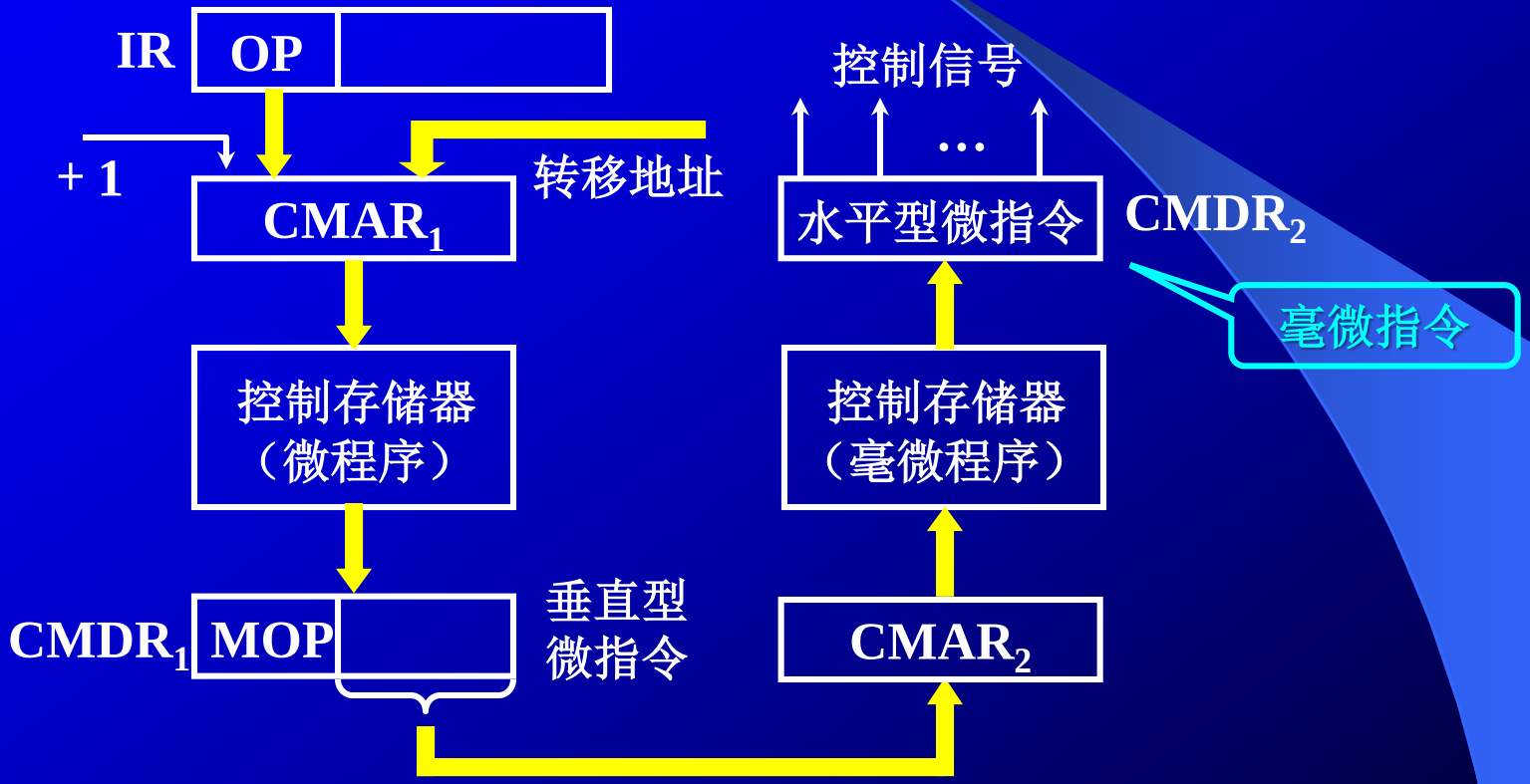
微程序设计　用 微程序解释机器指令

毫微程序设计　用 毫微程序解释微程序

毫微指令与微指令 的关系好比 微指令与机器指令 的关系

2. 毫微程序控制存储器的基本组成

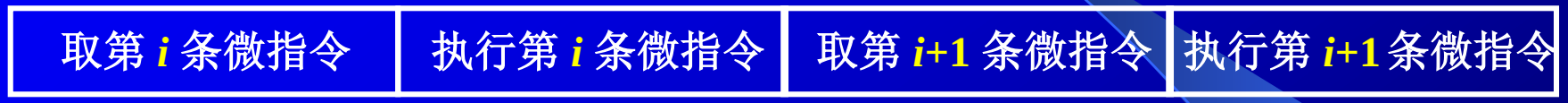
10.2



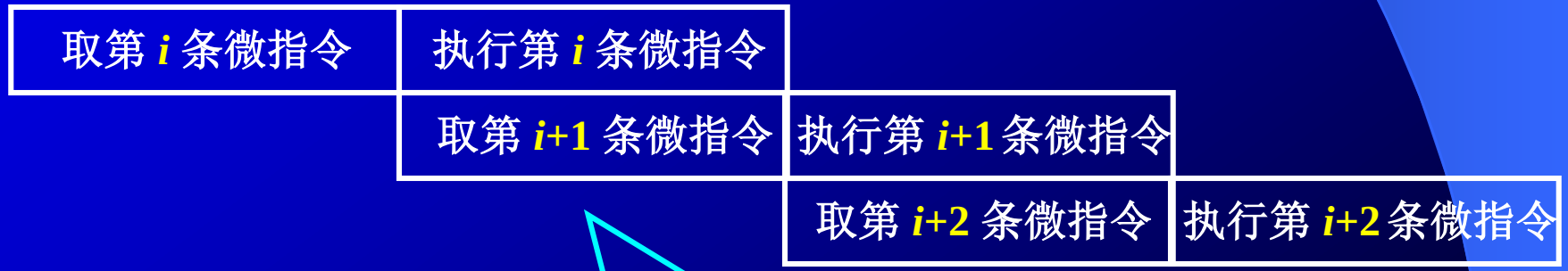
八、串行微程序控制和并行微程序控制

10.2

串行 微程序控制



并行 微程序控制



微指令预取
微指令流水

九、微程序设计举例

10.2

微程序设计控制单元的主要任务

- 编写对应各条机器指令的微程序
- 具体步骤是首先写出对应机器指令的全部微操作及节拍安排，然后确定微指令格式，最后编写出每条微指令的二进制代码 (称为微指令码点)

九、微程序设计举例

10.2

1. 写出对应机器指令的微操作及节拍安排

假设CPU结构与组合逻辑设计时相同，
要实现10条指令，且无间址和中断周期

(1) 取指阶段微操作分析　　3 条微指令

T_0　PC ⟶ MAR　　1 ⟶ R

T_1　M (MAR) ⟶ MDR　　(PC) + 1 ⟶ PC

T_2　MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

若一个 T 内安排一条微指令

则取指操作需 3 条微指令

还需考虑 如何读出 这 3 条微指令？

Ad (CMDR) ⟶ CMAR

OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(2) 取指阶段的微操作及节拍安排

考虑到需要 形成后续微指令的地址

T_0 PC ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR　(PC)+1 ⟶ PC

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ IR　　OP (IR) ⟶ 微地址形成部件

T_5 OP (IR) ⟶ 微地址形成部件 ⟶ CMAR

(3) 执行阶段的微操作及节拍安排

10.2

考虑到需形成后续微指令的地址

应为“取指微程序的入口地址 M”
由机器指令的微程序的最后一条微指令下地址字段指出

- 非访存指令

① CLA 指令

T_0　0 ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

“取指微程序的入口地址 M”

② COM 指令

T_0　$\overline{AC}$ ⟶ AC

T_1　Ad (CMDR) ⟶ CMAR

③ SHR 指令

T_0　　L (AC) ⟶ R (AC)　　$AC_0 \longrightarrow AC_0$

T_1　　Ad (CMDR) ⟶ CMAR

④ CSL 指令

T_0　　R (AC) ⟶ L (AC)　　$AC_0 \longrightarrow AC_n$

T_1　　Ad (CMDR) ⟶ CMAR

⑤ STP 指令

T_0　　0 ⟶ G

T_1　　Ad (CMDR) ⟶ CMAR

- 访存指令

⑥ ADD 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 (AC) + (MDR) ⟶ AC

T_5 Ad (CMDR) ⟶ CMAR

⑦ STA 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ W

T_1 Ad (CMDR) ⟶ CMAR

T_2 AC ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ M (MAR)

T_5 Ad (CMDR) ⟶ CMAR

⑧ LDA 指令

T_0 Ad (IR) ⟶ MAR　　1 ⟶ R

T_1 Ad (CMDR) ⟶ CMAR

T_2 M (MAR) ⟶ MDR

T_3 Ad (CMDR) ⟶ CMAR

T_4 MDR ⟶ AC

T_5 Ad (CMDR) ⟶ CMAR

- 转移类指令

⑨ JMP 指令

T_0　　Ad (IR) ⟶ PC

T_1　　Ad (CMDR) ⟶ CMAR

⑩ BAN 指令

T_0　　$A_0 \cdot Ad(IR) + \overline{A_0} \cdot (PC) \longrightarrow PC$

T_1　　Ad (CMDR) ⟶ CMAR

全部微操作　20种

微指令　38条

全部微操作　20种

微指令　38条

PC→MAR 微操作

1→R 微操作

M(MAR)→MDR

(PC) +1→PC

MDR→IR

0→AC

$\overline{AC}$→AC

L(AC)→R(AC),AC_0→AC_0

R(AC)→L(AC) ,AC_0→AC_n

0→G

Ad(IR)→MAR

(MDR) + (AC)→AC

1→W

AC→MDR

MDR→M(MAR)

MDR→AC

Ad(IR)→PC

$A_0 \cdot Ad(IR) + \overline{A}_0 \cdot (PC) \rightarrow PC$

Ad(CMDR)→CMAR

OP(IR) →微地址形成部件→CMAR

前6页幻灯片：
一个*T*，一个微指令

问题：
控制存储器里面有多少个微程序？

2. 确定微指令格式

(1) 微指令的编码方式

采用直接控制

(2) 后续微指令的地址形成方式

由机器指令的操作码通过微地址形成部件形成

由微指令的下地址字段直接给出

(3) 微指令字长

由 20 个微操作

确定 操作控制字段　　最少 20 位

由 38 条微指令

确定微指令的 下地址字段 为 6 位

微指令字长 可取 20 ＋ 6 ＝ 26 位

(4) 微指令字长的确定

38 条微指令中有 19 条

是关于后续微指令地址 ⟶ CMAR

其中 { 1 条　OP (IR) ⟶ 微地址形成部件 ⟶ CMAR
　　 { 18 条　Ad (CMDR) ⟶ CMAR

若用 Ad (CMDR) 直接送控存地址线

则 省去了输至 CMAR 的时间，省去了 CMAR

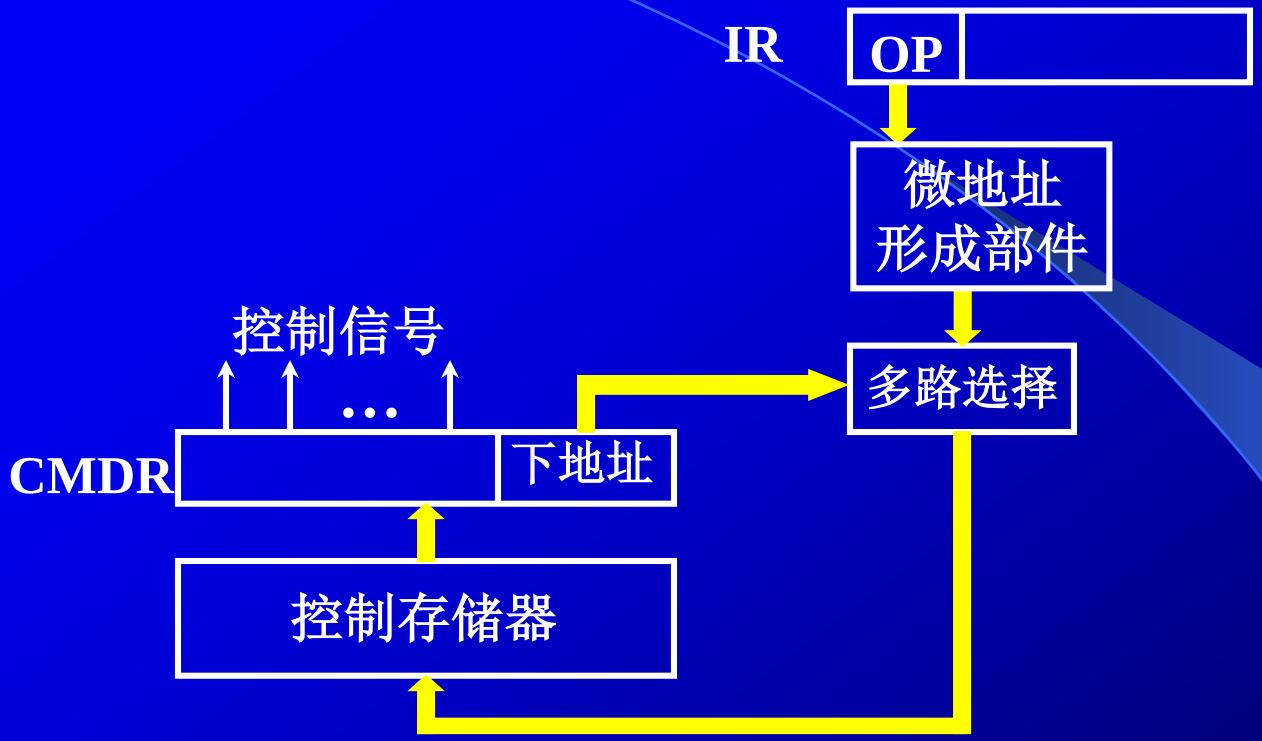
同理　OP (IR) ⟶ 微地址形成部件 ⟶ 控存地址线

可省去 19 条微指令，2 个微操作

38 － 19 ＝ 19　　20 － 2 ＝ 18

下地址字段最少取 5 位　　操作控制字段最少取 18 位

(5) 省去了 CMAR 的控制存储器



考虑留有一定的余量

取操作控制字段　18 位→24 位
下地址字段　5 位→ 6 位
} 共 30 位

(6) 定义微指令操作控制字段每一位的微操作

3. 编写微指令码点

微程序名称	微指令地址(八进制)	微指令（二进制代码）													
		操作控制字段									下地址字段				
		0	1	2	3	4	…	10	…	23	24	25	26	27	28	29
取指	00	1	1								0	0	0	0	0	1
	01			1	1						0	0	0	0	1	0
	02					1					×	×	×	×	×	×
CLA	03										0	0	0	0	0	0
COM	04										0	0	0	0	0	0
ADD	10		1					1			0	0	1	0	0	1
	11			1							0	0	1	0	1	0
	12										0	0	0	0	0	0
LDA	16		1					1			0	0	1	1	1	1
	17			1							0	1	0	0	0	0
	20										0	0	0	0	0	0

Summary Evaluation of Microprogramming

- **Pros**
  - systematic control unit design
  - ease of implementing a code-compatible family of computers
  - ability to emulate other computer systems
  - ability to debug system using microdiagnostics
  - field modification without rewiring or board replacement
  - low marginal cost for additional function, given that additional microinstructions fit within the control store or do not otherwise impact the packaging constraints
  - reconfigurable computing using a writable control store
- **Cons**
  - uneconomical for small systems due to relatively high fixed cost of control store and microinstruction sequencing logic (大型系统才能抹平成本)
  - another level of interpretation with associated overhead (译码增加开销)
  - instruction cycle time limited by control store access time (控存访问开销)
  - limited support tools, such as micro-assemblers, linkers, loaders, and machine simulators (which may provide various forms of debugging and tracing facilities) (欠缺编译开发工具链)

’00-’08 Trends Related to Microprogramming

- **Trends of application-specific processors, HW/SW co-design, code compression, and low-power designs**
  - **Tensilica Instruction Extension** language allows users to define application-specific instructions and user state registers in a subset of Verilog [Gon00]. This approach can be compared to the user-microprogrammable machines of the 1970s and 1980s. The Tensilica design process then implements the extended processor in silicon.
  - **UC Irvine’s NISC (No Instruction Set Computer)** is an approach that compiles C programs to microcode [Res05].
  - **Univ. of Pittsburgh’s PRISA (Post-manufacturing Reconfigurable ISA)** provides a generic datapath along with a large control store ROM from which the user can dynamically select an application instruction set of fifteen 16-bit instructions for decoding by a programmable instruction decoder [Che08]. The programmable instruction decoder is essentially a small writable control store (WCS) that can contain copies of a subset of the larger control store's lines.
  - **Stanford’s ELM project** replaces a traditional instruction cache with a control and index memory (CIM) and instruction register files (IRF) [Bla08]. The CIM/IRF arrangement is comparable to a two-level control store.

Mark Smotherman, https://people.cs.clemson.edu/~mark/hist.html

DPU from Omnitek
(acquired by Intel, Apr. 16, 2019)

Omnitek Deep Learning Processing Unit (DPU) achieves 135 GOPS/W at full 32-bit floating point accuracy when running the VGG-16 CNN in an Arria 10 GX 1150

Omnitek Launches the DPU for Intel FPGAs

Software Development | Application Execution | Hardware Optimization

TensorFlow | Caffe | OpenVINO | C/C++ Programming | python

Network Topology

Trained Weights

DPU SDK

Microcode Compiler

Weight Quantizer

SDRAM

Weights

Microcode

DPU Processing Engine

Omnitek IP

PCIe DMA

FPGA

DPU Driver

DPU API

Weights

Microcode

User Application

Control CPU

CNN | RNN | MLP | Future

Number of Engines (1, 2, 3, 4...
INT[4-16], FP[8-16]
DSP Array Size (256, 512, 1024...
Memory (Internal / External)
PCIe, DMA or SoC FPGA
Non-Linear Function
Elwise Support
FC Layer Support
Batch Processing
Sparse Network Optimisation

Omnitek IP Library

Camera ISP
Video Processing

HDMI 2.0
DisplayPort
SDI
...

Omnitek Design Services with IP

FPGA-based Processing Unit for ML which achieves World-Class Performance per Watt

作业

- 习题：10.15，10.21
- 本次作业提交截止时间：
  - 请于5月20日上课前提交
- 复习教材第10章，预习第4章

Q & A ?


中国科学院大学
University of Chinese Academy of Sciences
ICT
中科院计算所
INSTITUTE OF COMPUTING TECHNOLOGY,CAS