计算机组成

1 微机

微机结构

- Input 输入
- Output 输出
- Memory 存储器
- ALU 算术逻辑单元
- Control unit 控制单元

指令集：CISC(1-n个字)，RISC(1个字)

字：CPU一次可以处理的最大比特数

位扩展：同一地址的位扩展，满足一个字的输出

字扩展：增大字的量，选择不同的字，满足存储量需求

2 存储与I/O

内存特征

Location CPU,内部,外部

Capacity 字大小,字数目

Unit of transfer 内部(一字),外部(多字) — Addressable Unit: 内部(一字节),外部(簇)

Access method 访存方式

- Sequential 串行访问 (tape)
- Direct 直接访问 (disk)
- Random 随机访问 (RAM,ROM)
- Associative 关联访问 (cache)

Performance 评价指标

- Access time
- Memory Cycle time
- Transfer Rate

Physical type 物理类型

- Semiconductor (RAM)
- Magnetic (disk & tape)
- Optical (CD & DVD)
- Others (Bubble Hologram)

Organisation Physical arrangement of bits into words(存储字)

I/O数据传送方式

程序控制方式 Programmed I/O CPU与外设之间的数据传送是在程序控制下完成的。用查询方式使CPU与外设交换数据时，CPU要不断读取状态位，检查输入设备是否已准备好数据。由于许多外设的速度很低，这种等待过程会占去CPU的绝大部分时间，而真正用于传输数据的时间却很少，使CPU的利用率变得很低。

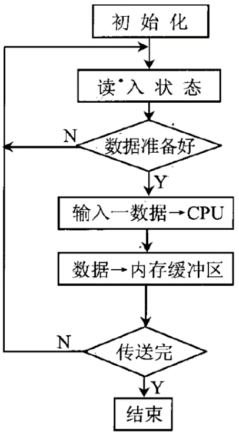


图 6.7 查询式输入流程图

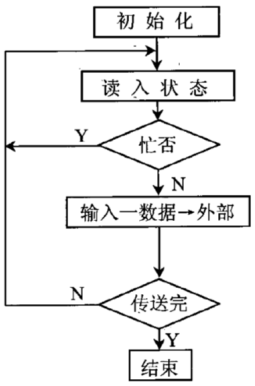


图 6.9 查询式输出流程图

中断方式 Interrupt driven I/O 采用中断方式后，CPU平时可以执行主程序，只有当输入设备将数据准备好了，或者输出端口的数据缓冲器已空时，才向CPU发中断请求。CPU响应中断后，暂停执行当前的程序，转去执行管理外设的中断服务程序(ISR)。在中断服务程序中，用于输入或输出指令在CPU和外设之间进行一次数据交换。等输入或输出操作完成之后，CPU又回去执行原来的程序。

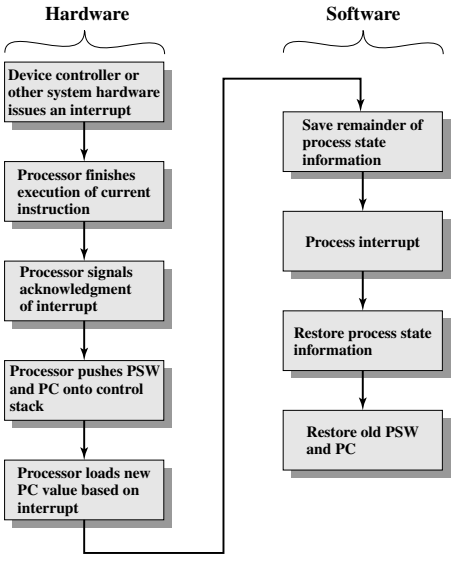


Figure 7.6 Simple Interrupt Processing

直接存储器访问 DMA DMA控制器临时接管总线，控制外设和存储器之间进行高速的数据传送，快速完成交换一批数据的任务，而不要CPU进行干预。这种控制器能给访问内存所需要的地址信息，并且能够自动修改地址指针，也能够设定和修改传送的字节数，还能够向存储器和外设发出相应的读写控制信号。在DMA传送结束后，它能够释放总线，把对总线的控制权又交还给CPU。

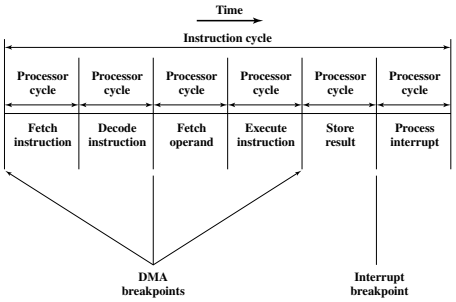


Figure 7.12 DMA and Interrupt Breakpoints during an Instruction Cycle

3 80x86

8086 16-bit, 20-bit address.

8088 16-bit internal, 8-bit external.

只有两段:

BIU (Bus Interface Unit) 连接内存与外设。

EU (Execution Unit) 执行之前获取的指令。

双工作模式

最小模式 MN/$\overline{\text{MX}}$ = 1 单 CPU。

最大模式 MN/$\overline{\text{MX}}$ = 0 多CPU(8086+8087)，8288控制芯片。

8086读周期时序: 在8086读周期内,有关总线信号的变化如下:

1. M/IO在整个读周期保持有效,当进行存储器读操作时,M/IO为高电平;当进行I/O端口读操作时,M/IO为低电平.
2. A19/S6～A16/S3是在T1期间,输出CPU要读取的存储单元的地址高4位.T2～T4期间输出状态信息S6～S3.
3. BHE/S7在T1期间输出BHE有效信号(BHE为低电平),表示高8位数据总线上的信息可以使用,BHE信号通常作为奇地址存储体的选择信号(偶地址存储体的选择信号是最低地址位A0).T2～T4期间输出高电平.
4. AD15～AD0在T1期间输出CPU要读取的存储单元或I/O端口的地址A15～A0.T2期间为高阻态,T3～T4期间,存储单元或I/O端口将数据送上数据总线.CPU从AD15～AD0上接收数据.

表 1: 微机概念差异

微处理器 Microprocessor	可以被微缩成集成电路规模的CPU电路，包含ALU,CU,寄存器
微型计算机 Mircrocomputer	微处理器,存储器,I/O,总线
微型计算机系统 Microcomputer system	以微型计算机为主体，配上I/O及系统软件就构成了微型计算机系统。
微控制器 Microcontrollers	A microcontroller has a CPU in addition to a fixed amount of RAM, ROM, I/O ports on one single chip (*e.g.* Cortex)
嵌入式系统 Embedded Systems	An embedded system uses a microcontroller or a microprocessor to do one task and one task only

表 2: 总线类型

类型	仲裁	时序
单工	集中式	同步
多工	分布式	异步

表 3: 总线结构

	优点	缺点
单线结构	简单	吞吐量低
CPU-Central 双线结构	数据传输率高	I/O与内存需要经过CPU
Memory-Central 双线结构	CPU性能好 吞吐量高	

表 4: RAM 区别

	DRAM	SRAM
字节存储方式	电容电荷	开关状态
电荷泄漏	有	无
刷新	需要	不需要
构造	简单	复杂
每位规模	更小	更大
价格	便宜	昂贵
刷新电路	需要	不需要
速度	更慢	更快
用途	主存储器	高速缓存

表 5: ROM 区别

掩膜型 ROM	无法修改
可编程只读存储器 PROM	一旦写入，不可改变
可擦除可编程只读存储器 EPROM	可写，用紫外光擦除，重新写入
点可擦除的可编程只读存储器 EEPROM	通电擦除，重新写入
闪存 Flash	对芯片编程，通电擦除再写入

表 6: DMA 架构

	使用总线次数	CPU暂停次数
Single Bus, Detached	2	2
Single Bus,Intergrated	1	1
Seperate I/O	1	1

表 7: 8086 寄存器

Category	Bits	Register Names
General	16	`AX`(Accumulator), `BX`(Base), `CX`(Count), `DX`(Data)
	8	`AH`, `AL`, `BH`, `BL`, `CH`, `CL`, `DH`, `DL`
Pointer	16	`SP`(Stack Pointer), `BP`(Base Pointer)
Segment	16	`CS`(Code Segment), `DS`(Data Segment), `SS`(Stack Segment), `ES`(Extra Segment)
Instruction	16	`IP`(Instruction Pointer)
Flag	16	`FR`(Flag Register)

表 8: 段偏移寄存器

Segment register	`CS`	`DS`	`ES`	`SS`
Offset register	`IP`	`SI,DI,BX`	`SI,DI,BX`	`SP,BP`

表 9: Flag 寄存器

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R	R	R	OF	DF	IF	TF	SF	ZF	U	AF	U	PF	U	CF

5. ALE:在T1期间地址锁存有效信号,为一正脉冲,系统中的地址锁存器正是利用该脉冲的下降沿来锁存A19/S6～A16/S3,ADl5～AD0中的20位地址信息以及BHE.
6. RD在T2期间输出低电平,送到被选中的存储器或I/O接口.要注意的是,只有被地址信号选中的存储单元或I/O端口,才会被RD信号从中读出数据(数据送上数据总线ADl5～AD0).
7. DT/R在整个总线周期内保持低电平,表示本总线周期为读周期.在接有数据总线收发器的系统中,用来控制数据传输的方向.
8. DEN在T2～T3期间输出有效低电平,表示数据有效.在接有数据总线收发器的系统中,用来实现数据的选通.

8086写周期时序总线写操作的时序与读操作时序相似,其不同处在于:

1. AD15～AD0在T2～T4期间送上欲输出的数据,而无高阻态.
2. WR在T2～T4期间输出有效低电平,该信号送到所有的存储器和I/O接口.要注意的是,只有被地址信号选中的存储单元或I/O端口才会被WR信号写入数据.
3. DT/R在整个总线周期内保持高电平,表示本总线周期为写周期.在接有数据总线收发器的系统中,用来控制数据传输方向.

条件符:

CF (Carry Flag): 进位符 set whenever there is a carry out, from d7 after a 8-bit op, from d15 after a 16-bit op

PF (Parity Flag): 校验符 the parity of the op result's low-order byte, set when the byte has an even number of 1s （偶1为1）

AF (Auxiliary Carry Flag): 辅助进位符 set if there is a carry from d3 to d4, used by BCD-related arithmetic

ZF (Zero Flag): 零 set when the result is zero

SF (Sign Flag): 符号符 copied from the sign bit (the most significant bit) after op

OF (Overflow Flag): 溢出符 set when the result of a signed number operation is too large, causing the sign bit error

控制符:

IF (Interrupt Flag): 中断符 set or cleared to enable or disable only the external maskable interrupt requests

> After reset, all flags are cleared which means you (as a programmer) have to set IF in your program if allow INTR.

```
CLI ;clears IF
STI ;sets IF
```

DF (Direction Flag): 方向符 indicates the direction of string operations

TF (Trap Flag): 陷阱符 when set it allows the program to single-step, meaning to execute one instruction at a time for debugging purposes

4 汇编

表 12: MODEL 大小

.MODEL	code～64KB	data～64KB
SMALL	>	≤
MEDIUM	>	≤
COMPACT	≤	>
LARGE	>	><
HUGE	>	>
TINY	<	

PTR 可以暂时性更改一个变量的类型。

```
DATA1  DB  10H,20H,30H  ;
DATA2  DW  4023H,0A845H

MOV  BX, WORD PTR DATA1
; 2010H -> BX
MOV  AL, BYTE PTR DATA2
; 23H -> AL
MOV  WORD PTR [BX], 10H
; [BX],[BX+1] <- 0010H

JMP FAR PTR aLabel
```

8086 中没有内存到内存的直接运算。

除法运算比较特殊。

type	op1	op2	$\frac{op1}{op2}$	mod
B/B	AL	src	AL	AH
W/W	AX	src	AX	DX
W/B	AX	src	AL	AH
DW/W	DX,AX	src	AX	DX

跳转前的比较：`CMP dest,src`

	CF	ZF
dest > src	0	0
dest = src	0	1
dest < src	1	0

其他跳转:

Mnemonic	Condition	跳转条件
JNC	CF=0	不进位
JNE/JNZ	ZF=0	不等于0
JNO	OF=0	没有溢出
JNP/JPO	PF=0	奇数个1
JNS	SF=0	不是负数
JP	OF=1	溢出
JP/JPE	PF=1	偶数个1
JS	SF=1	负数

5 芯片

5.1 存储器

Checksum 不进位地将所有的字节相加，取和的二的补码。

Parity bit 考虑原有位和奇偶校验位1的个数的总和。偶校验：当序列中有奇数个1时设置为1；奇校验：当序列中有偶数个1时设置为1。

CRC for disks and the Internet. k-bit data, n-bit CRC:

$$CRC = (M(X) \times X^n) \bmod G(X) \quad (1)$$

5.2 外设 8255

5.3 计时器 8253

5.4 中断 8259

8086/8088 有 256 个中断类型。

```
INT 00H
INT 0FFH
```

中断向量表(IVT)

```
CS = Type * 4
IP = Type * 4 + 2
```

中断服务程序(ISR)

	CALL FAR	INT
jump	anywhere ≤ 1MB	a fix location
occurence	in the sequence of instructions	an external interrupt at any time
maskable	cannot be masked (disabled)	can be masked
save	CS:IP	FR+CS:IP
return	RETF	IRET

中断优先级:

$$\text{INT} > \text{NMI} > \text{INTR} \quad (2)$$

高优先级中断（数字小的）可以打断低优先级中断，低优先级中断只能在 EOI 之后才能够打断高优先级中断。

```
STI (EOI) IRET
```

5.5 连接 8251

	Parrallel	**Serial**
transfer	每一位用一条线	只用一条线
bit	一般用8条以上的线	线上每次传一位
control signal	需要额外的控制信号	不需要
usage	快速 昂贵 短距离	便宜 慢速 长距离

表 10: 8086 接脚

Signal	**Description**	0	1
ALE	Address Latch Enabled		Latched
$\overline{\text{BHE}}$	Bank High Enabled	$AD_8 \sim AD_{15}$ Enabled	$AD_8 \sim AD_{15}$ Disabled
DT/$\overline{\text{R}}$	direction of Data Transfer	sending data	receiving data
$\overline{\text{DEN}}$	Data transceiver ENabled	enabled	disabled
$\overline{\text{WR}}$	WRiting to Mem/IO	writing	
$\overline{\text{RD}}$	ReaDing from mem/IO	reading	
M/$\overline{\text{IO}}$	CPU accessing Memory / IO	IO	Memory
INTR	INTerrupt Request, maskable by clearing IF		Requesting
$\overline{\text{INTA}}$	INTerrupt Acknowledge		Acknowledge
NMI	Non-Maskable Interrupt, CPU is interrupted after finishing the current instruction; cannot be masked by software		will be interrupted
HOLD	HOLD the bus request		hold
HLDA	HoLD request acknowledge		hold req ack
$\overline{\text{TEST}}$	for debug	test	
READY	mem/IO is READY for transfer		ready
RESET	reset the CPU, IP, DS, SS, ES and 6 inst in instruction queue are cleared		CS=0FFFFH

表 11: MOV 指令

Instruction	Segment Used	Default Segment
MOV AX, CS:[BP]	CS:BP	SS:BP
MOV DX, SS:[SI]	SS:SI	DS:SI
MOV AX, DS:[BP]	DS:BP	SS:BP
MOV CX, ES:[BX]+12	ES:BX+12	DS:BX+12
MOV SS:[BX][DI]+32, AX	SS:BX+DI+32	DS:BX+DI+32

表 13: PROC 范围

PROC	IP	current code segment
SHORT	-128∼127(±1B)	
NEAR	-32768∼32767(±2B)	within
FAR	changed along with CS	outside

CALL *procedure*
RET

表 14: 数据定义

	描述	**示例**
ORG	指定开始偏移量	ORG 10H
DB	分配字节大小的块，依次写入	Z DB "Good Morning"
EQU	定义常量	NUM EQU 234
DUP	复制字符	x DB 6 DUP(23H)

CMP op1,op2

表 15: 跳转指令

Mnemonic	Condition	op1∼op2
Unsigned		
JA/JNBE	CF=0 & ZF=0	$>$
JAE/JNB	CF=0	$\geq$
JB/JNAE	CF=1	$<$
JBE/JNA	CF=1 | ZF=1	$\leq$
Signed		
JG/JNLE	SF=OF & ZF=0	$>$
JGE/JNL	SF=OF	$\geq$
JL/JNGE	SF $\oplus$ OF	$<$
JLE/JNG	ZF=1 | SF $\oplus$ OF	$\leq$

表 16: 算术运算

	op1	,op2	calc
ADD	dest	src	dest = dest src
ADC	dest	src	dest = dest + src + CF
SUB	dest	src	dest = dest - src
SBB	dest	src	dest = dest - src - CF
INC	dest		dest = dest + 1
DEC	dest		dest = dest - 1
MUL	src		(DX,) AX = src * AX
DIV	src		AL = AL / src, AH = AL mod src*

表 17: 逻辑运算

	op1	,op2	calc
AND	dest	src	dest = dest & src
OR	dest	src	dest = dest | src
XOR	dest	src	dest = dest $\oplus$ src
NOT	dest		dest = $\sim$dest
SHR	dest	reg	dest = dest >> reg(zero-ext)
SHL	dest	reg	dest = dest << reg
ROL	dest	reg	dest = >> dest << reg
RCL	dest	reg	dest = >> CF << dest << reg

表 18: 存储器访问

	$\overline{BHE}$	A0	访问
MOV AL, [100h]	1	0	D7-D0
MOV AL, [101h]	0	1	D15-D8
MOV AX, [100h]	0	0	D15-D0
MOV AX, [101h]	0	1	D15-D8
	1	0	D7-D0

表 19: 8255 功能信号

$\overline{CS}$	A_1	A_0	$\overline{RD}$	$\overline{WR}$	**Function**
0	0	0	0	1	PA→Data bus
0	0	1	0	1	PB→Data bus
0	1	0	0	1	PC→Data bus
0	0	0	1	0	Data bus→PA
0	0	1	1	0	Data bus→PB
0	1	0	1	0	Data bus→PC
0	1	1	1	0	Data bus→CR
1			1	1	$D_0 \sim D_7$ in float

表 20: 8255 模式

	Mode 0	**Mode 1**	**Mode 2**	**BSR**
PCU	$PC_7 \sim PC_4$	$PC_7 \sim PC_3$	$PC_7 \sim PC_3$	
PCL	$PC_3 \sim PC_0$	$PC_2 \sim PC_0$		
PA	I/O	I/O	I&O	I
PB	I/O	I/O		I
PCU	I/O	Handshake for PA	Handshake for PA	O
PCL	I/O	Handshake for PB		

表 21: 8255 模式字(CR)

	D7	D6	D5	D4	D3	D2	D1	D0
	1	Group A	Group A	PA	PCU	Group B	PB	PCL
0		00 Mode 0 Simple I/O		Output	Output	Mode 0	Output	Output
1	Input/Output modes	1* Mode 2	01 Mode 1	Input	Input	Mode 1	Input	Input
	0	*	*	*	PC	PC	PC	Set
0	BSR mode				000 PC_0			Reset
1					111 PC_7			Set

表 22: 8253 功能信号

$\overline{CS}$	A_1	A_0	$\overline{RD}$	$\overline{WR}$	**Function for counter**
0	0	0	1	0	W C0
0	0	1	1	0	W C1
0	1	0	1	0	W C2
0	1	1	1	0	W CP
0	0	0	0	1	R C0
0	0	1	0	1	R C1
0	1	0	0	1	R C2
0	1	1	0	1	*R CP (for 8254)*
1			*	*	Not Available

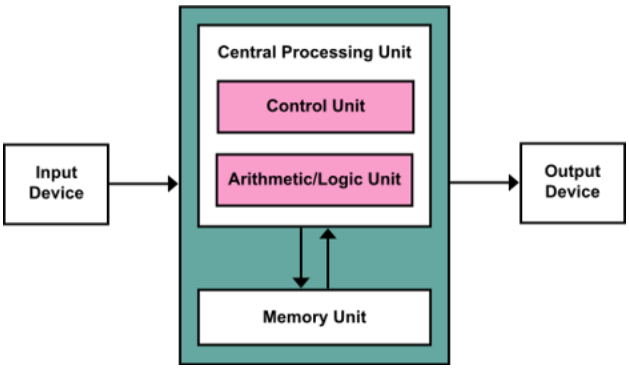
表 23: 8253 模式

	Mode 0	Mode 1	Mode 2	Mode 3	Mode 4	Mode 5
	Interrupt on terminal count	One-shot	Rate-generator	Square wave rate-generator	Software triggered strobe	Hardware triggered strobe
initial	L	H	H	H	H	H
count started	L	L	H	H	H	H
Gate=1	enable counting	pulse to retrigger	repeat counting	repeat counting	enable	pulse to retrigger
Gate=0	disable counting	enable	disable counting	Output L→H, disabled	disable	enable
Terminal	H	H	L	odd: H for $\frac{n+1}{2}$, L for $\frac{n-1}{2}$	L→H	L→H

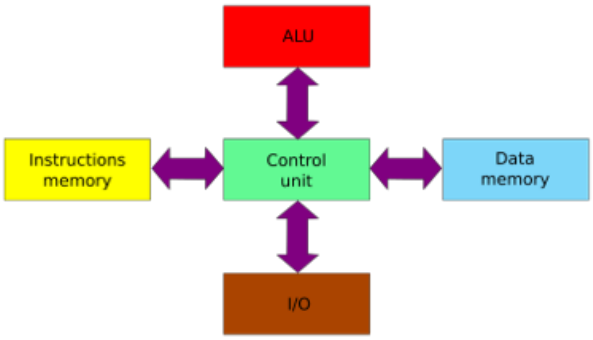
表 24: 8253 模式字

D7	D6		D5	D4		D3	D2	D1		D0	
SC1	SC0		RW1	RW0		M2	M1	M0		BCD	
0	0	Select Counter 0	0	0	Counter latch command	0	0	0	Mode 0	0	16-bit
0	1	Select Counter 1	0	1	R/W least byte	0	0	1	Mode 1	1	BCD
1	0	Select Counter 2	1	0	R/W most byte	*	1	0	Mode 2		
1	1	*READ back for 8254*	1	1	R/W least, then most	*	1	1	Mode 3		
						1	0	0	Mode 4		
						1	0	1	Mode 5		

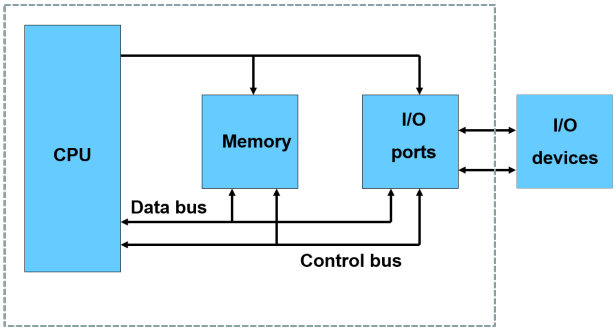
冯诺依曼结构



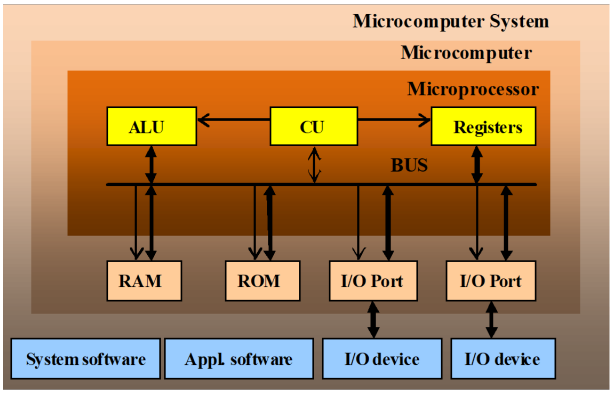
哈佛结构



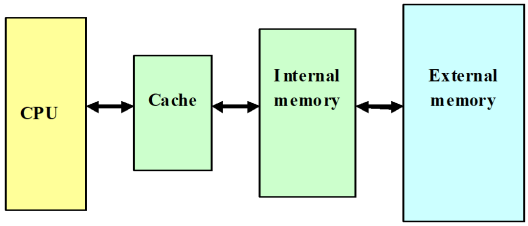
微机结构



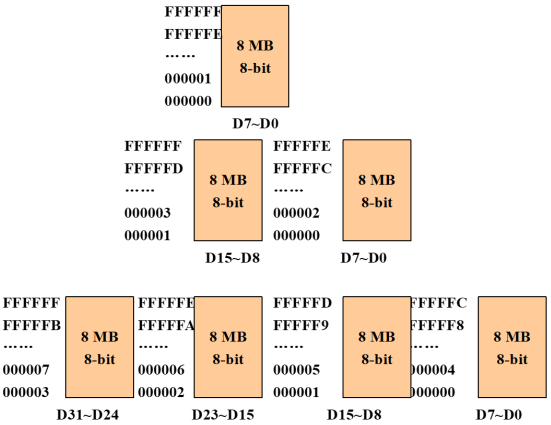
微机系统结构（哈佛结构）



内存层级

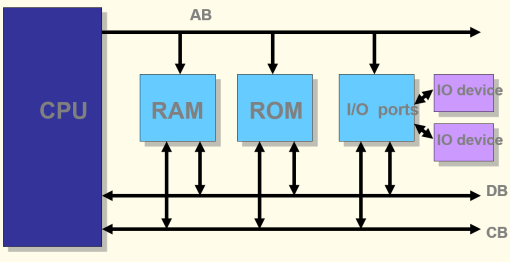


内存组织

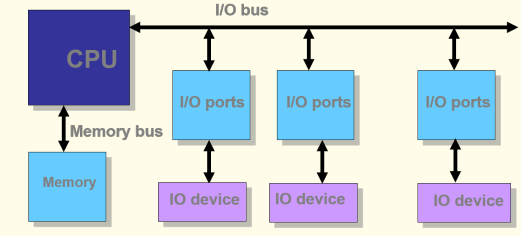


总线结构

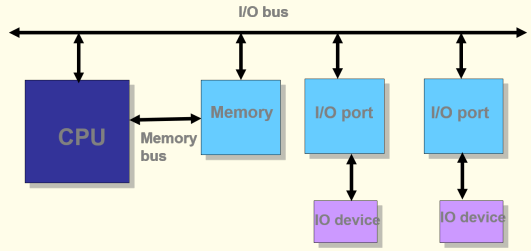
单线结构



CPU-Central 双线结构

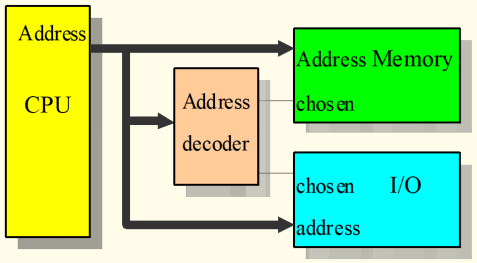


Memory-Central 双线结构



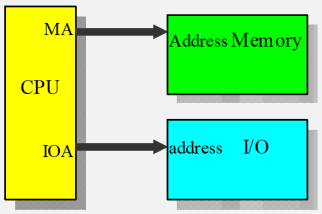
寻址方式

存储器映像寻址方式

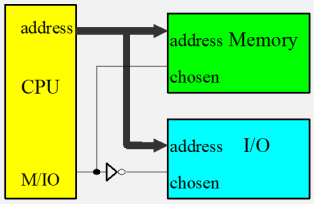


I/O单独编址方式

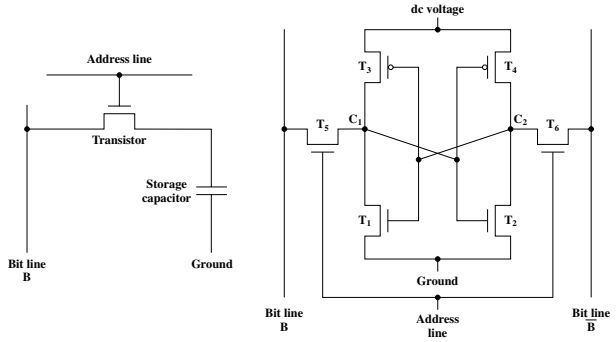
单工地址线



多工地址线



RAM



(a) Dynamic RAM (DRAM) cell (b) Static RAM (SRAM) cell

4M×4 DRAM

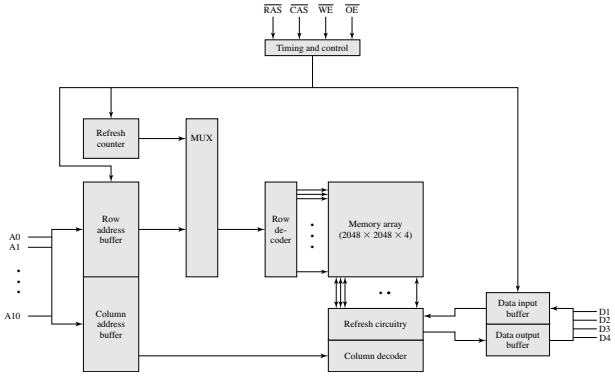
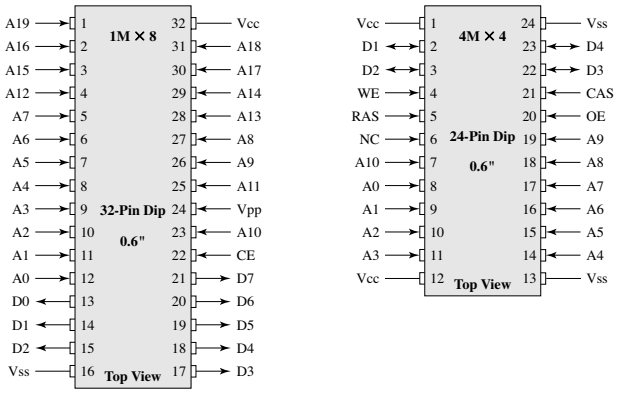


Figure 5.3 Typical 16 Megabit DRAM (4M × 4)

存储芯片封装



(a) 8-Mbit EPROM (b) 16-Mbit DRAM

存储位扩展

256K×8-bit: 8 256×1-bit

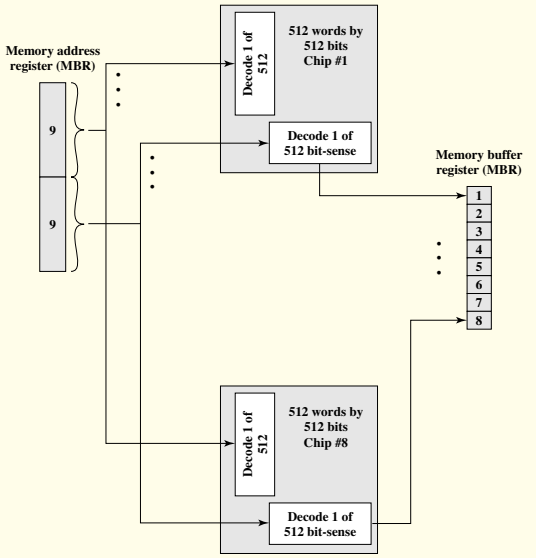
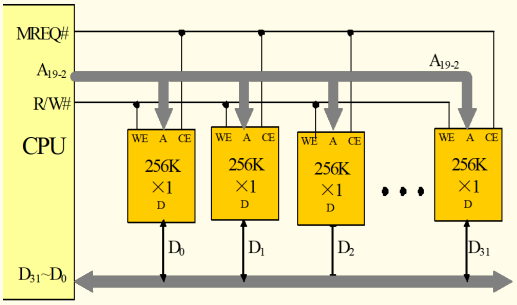


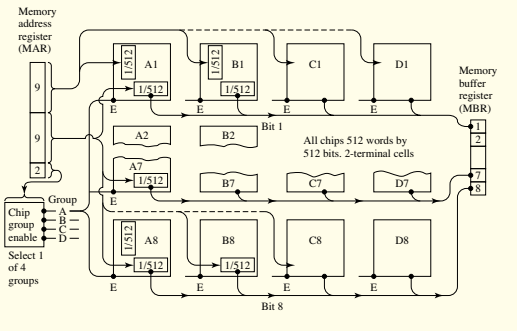
Figure 5.5 256-KByte Memory Organization

256K×32-bit: 32 256K×1-bit

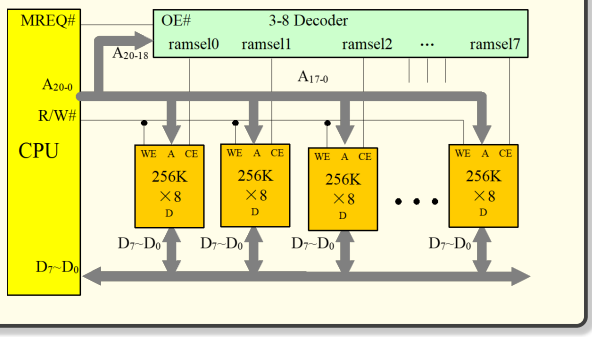


存储字扩展

1M×8-bit: 4 group 256K×1-bit

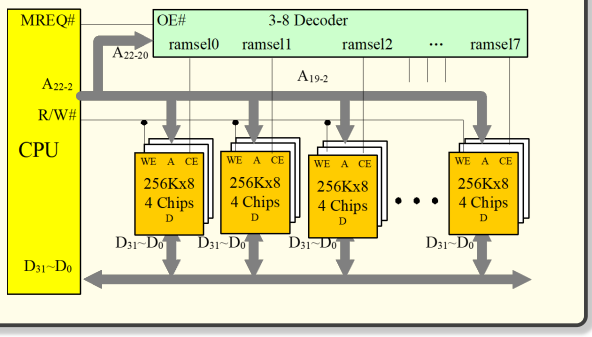


2M×8-bit: 8 group 256K×8-bit

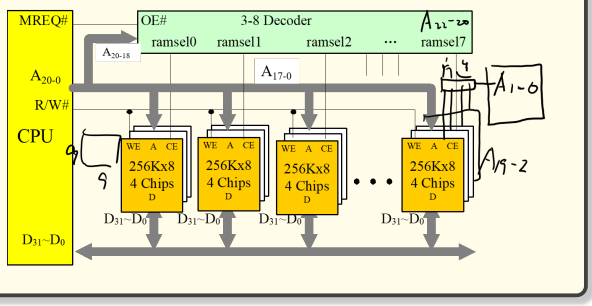


字与位同时扩展

32位可寻址单元



1字节可寻址单元



外设

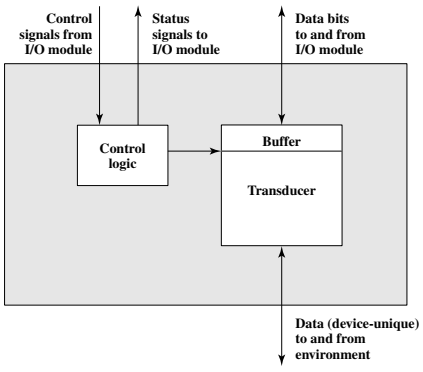


Figure 7.2 Block Diagram of an External Device

I/O模块

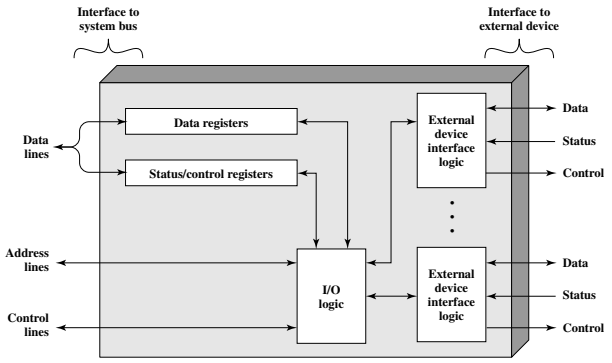


Figure 7.3 Block Diagram of an I/O Module

中断处理内存变化

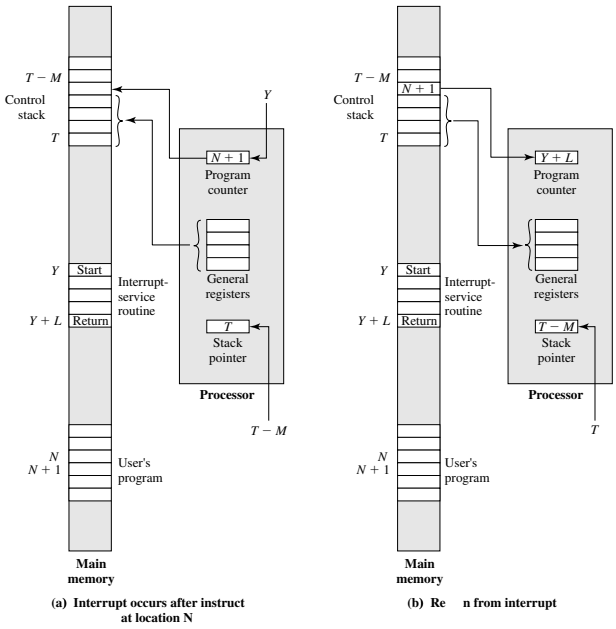


Figure 7.7 Changes in Memory and Registers for an Interrupt

DMA架构

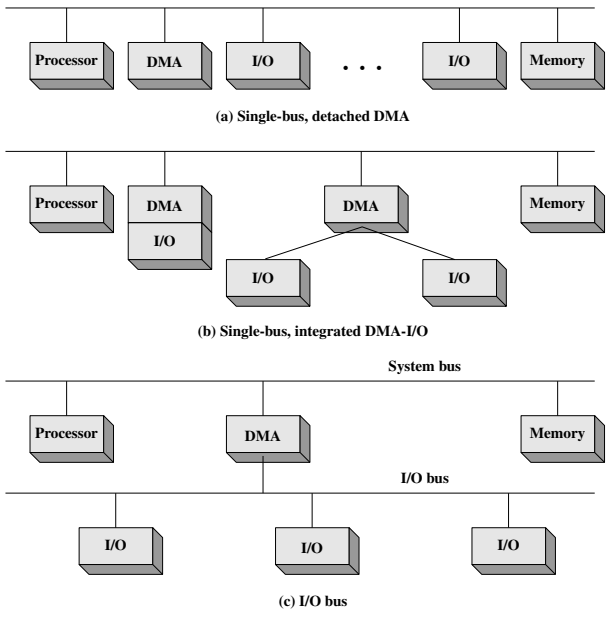
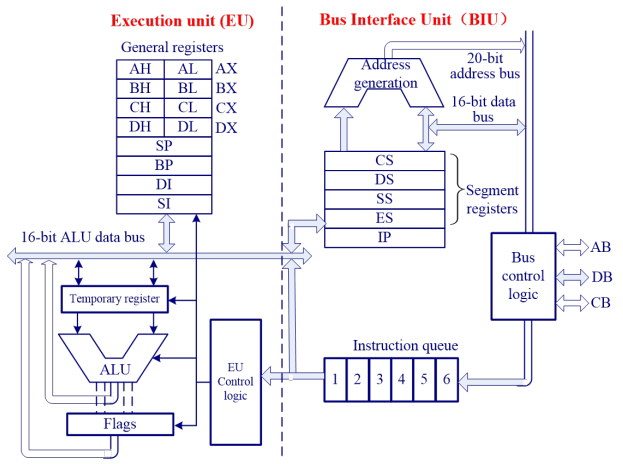
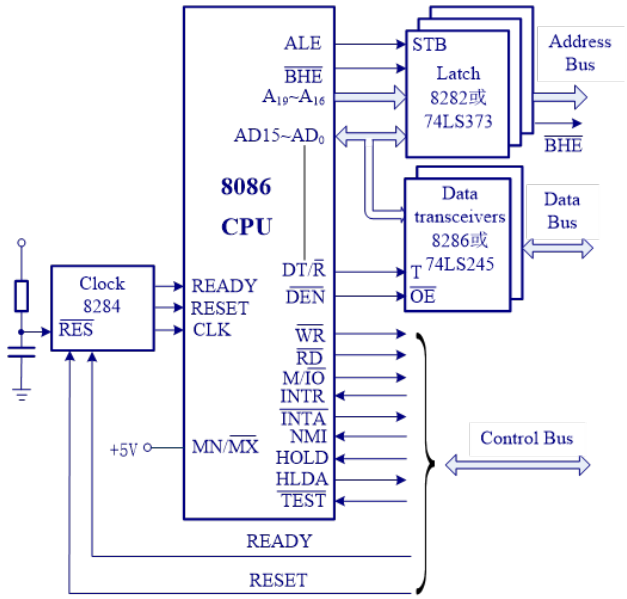


Figure 7.13 Alternative DMA Configurations

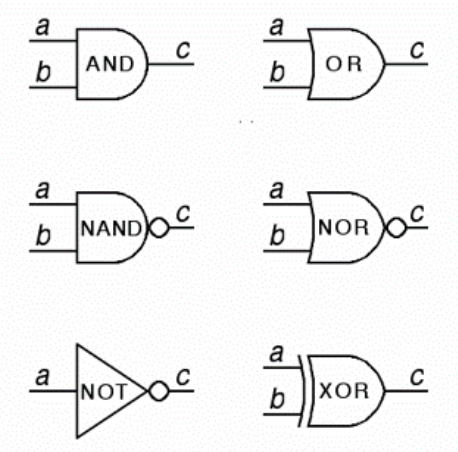
8086内部结构



8086配置图



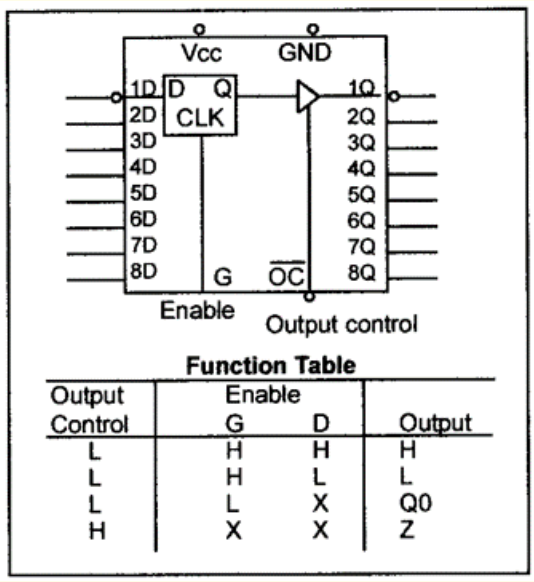
逻辑门电路



input		output					
a	b	ab	$a+b$	$\overline{ab}$	$\overline{a+b}$	$\overline{a}$	$a \oplus b$
0	0	0	0	1	1	1	0
0	1	0	1	1	0	1	1
1	0	0	1	1	0	0	1
1	1	1	1	0	0	0	0

锁存器和数据传输器

74LS373 D Latch



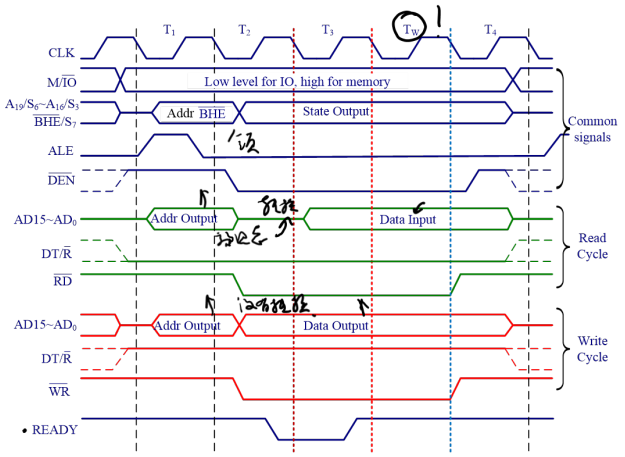
Function Table

Output Control	Enable G	D	Output
L	H	H	H
L	H	L	L
L	L	X	Q0
H	X	X	Z

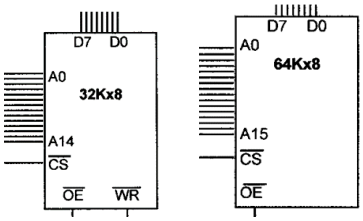
Data Bus Transceiver

INPUTS		OUTPUT
$\overline{E}$	DIR	
L	L	Bus B Data to Bus A
L	H	Bus A Data to Bus B
H	X	Isolation

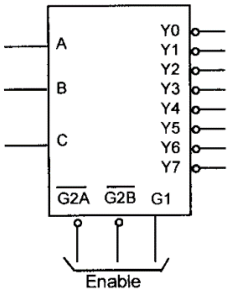
8086/88 总线周期



RAM&ROM



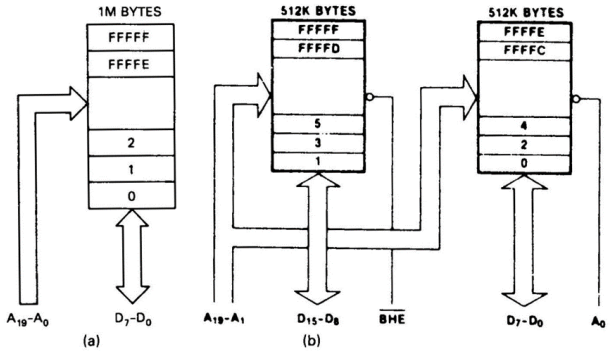
74LS138 38译码器



Function Table

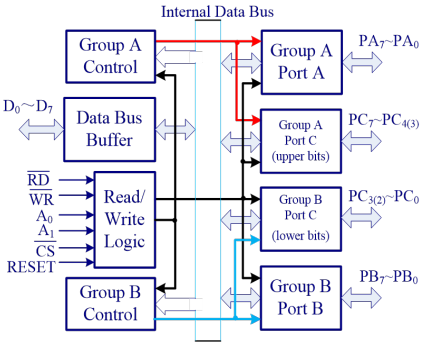
Inputs				Outputs							
Enable		Select									
G1	G2	C	B	A	Y0	Y1	Y2	Y3	Y4	Y5	Y6	Y7
X	H	X	X	X	H	H	H	H	H	H	H	H
L	X	X	X	X	H	H	H	H	H	H	H	H
H	L	L	L	L	L	H	H	H	H	H	H	H
H	L	L	L	H	H	L	H	H	H	H	H	H
H	L	L	H	L	H	H	L	H	H	H	H	H
H	L	L	H	H	H	H	H	L	H	H	H	H
H	L	H	L	L	H	H	H	H	L	H	H	H
H	L	H	L	H	H	H	H	H	H	L	H	H
H	L	H	H	L	H	H	H	H	H	H	L	H
H	L	H	H	H	H	H	H	H	H	H	H	L

8086存储组织

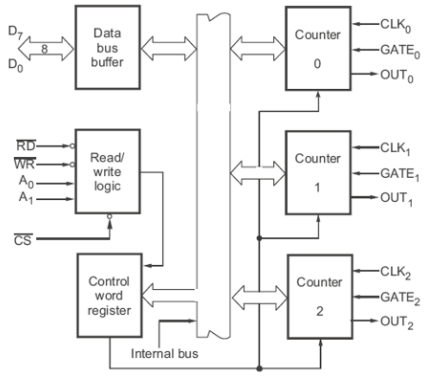


$\overline{BHE}$	A0		
0	0	Even word	D0-D15
0	1	Odd byte	D8-D15
1	0	Even byte	D0-D7
1	1	None	

8255 外设芯片

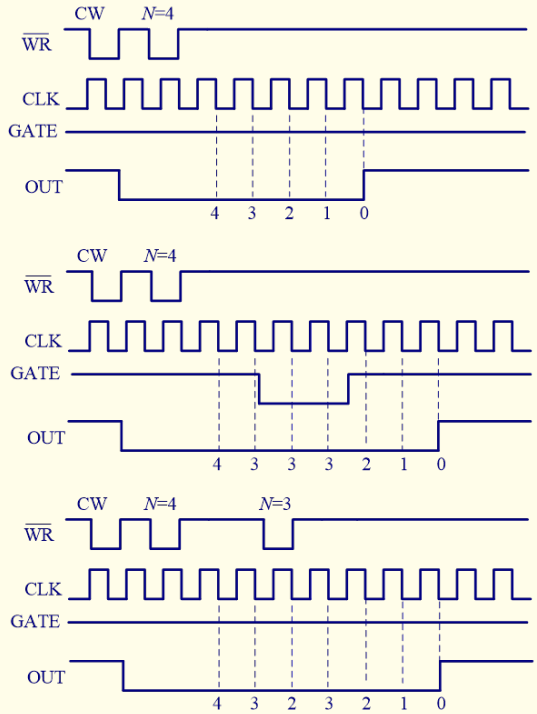


8253 计时器

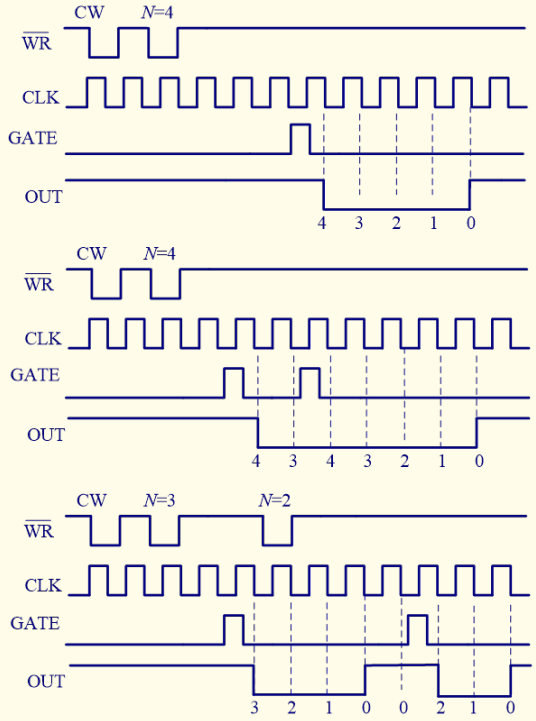


8253 计时器模式

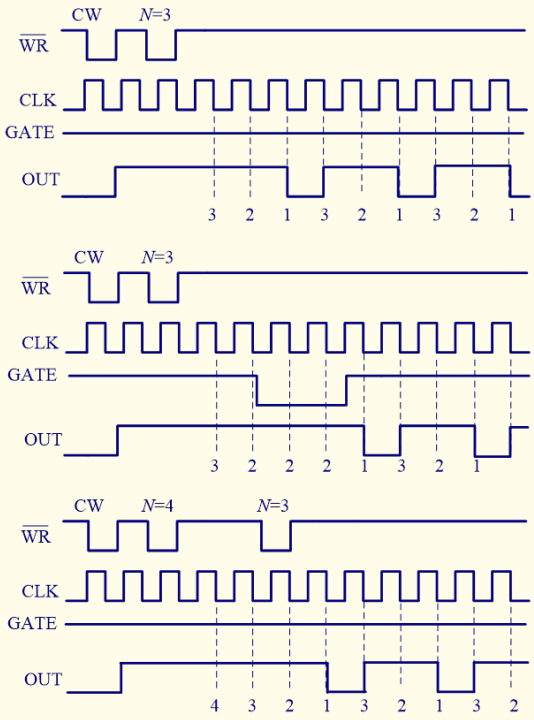
Mode 0—Interrupt on Terminal Count



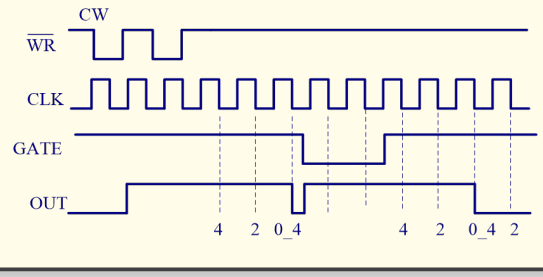
Mode 1—Hardware Retriggerable One-shot



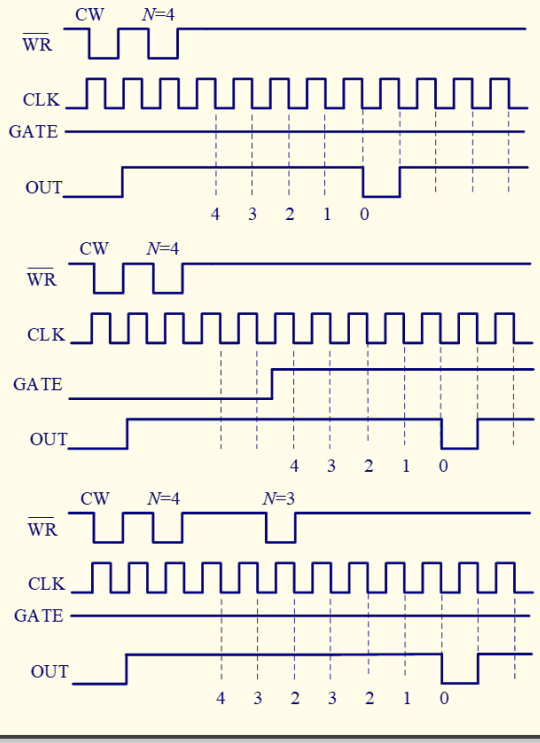
Mode 2—Rate Generator



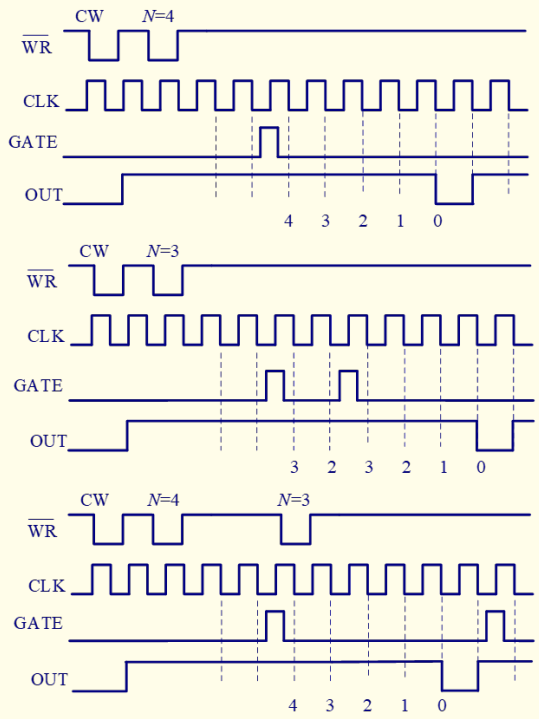
Mode 3—Square Wave Rate Generator



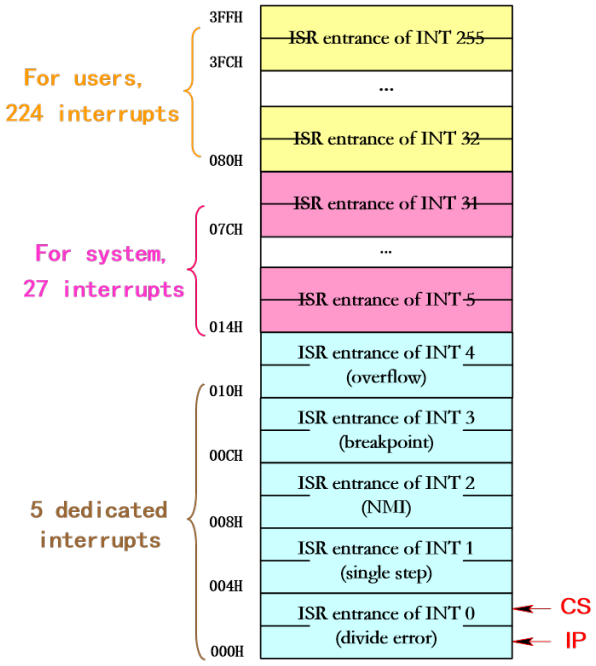
Mode 4—Software Triggered Strobe



Mode 5—Hardware Triggered Strobe (Retriggerable)



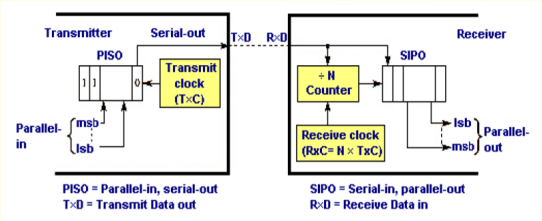
8086/88 中断向量表



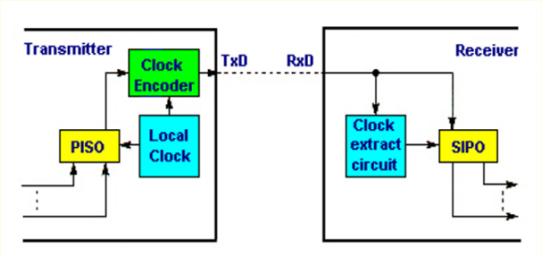
同步与异步

异步	同步
一次传一个字节	一次传一块数据
使用start bit将接发者同步	使用 synch characters 将接发者同步

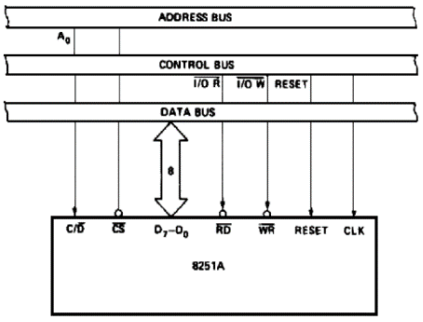
异步



同步



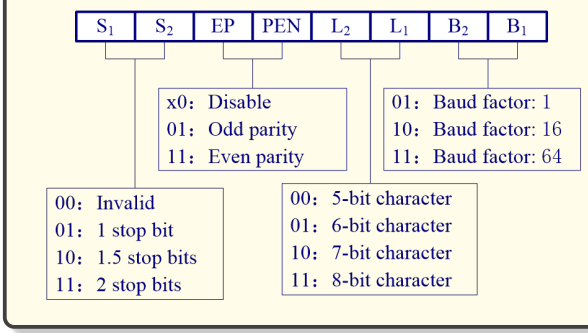
8251 接口



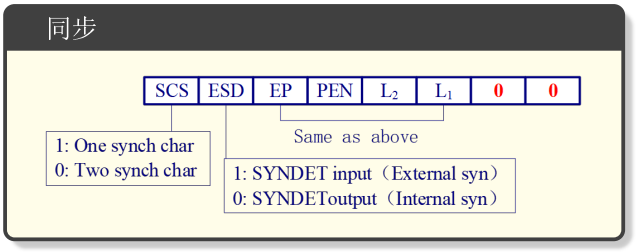
C/D	RD	WR	CS	Function
0	0	1	0	8251 Data → data bus
0	1	0	0	8251 Bus → 8251 data
1	0	1	0	status → data bus
1	1	0	0	data bus → control
*	1	1	0	data bus → 3-state
*	*	*	1	data bus → 3-state

8251 模式字

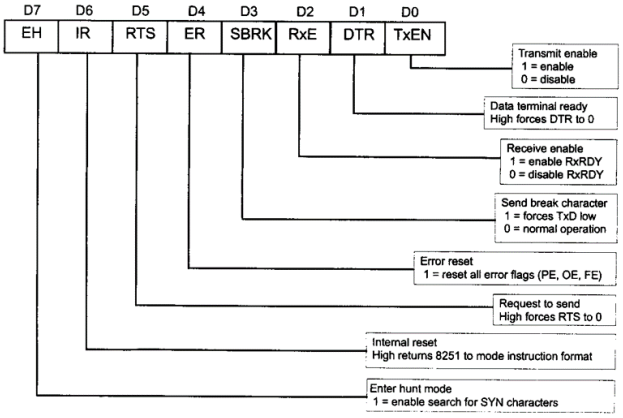
异步



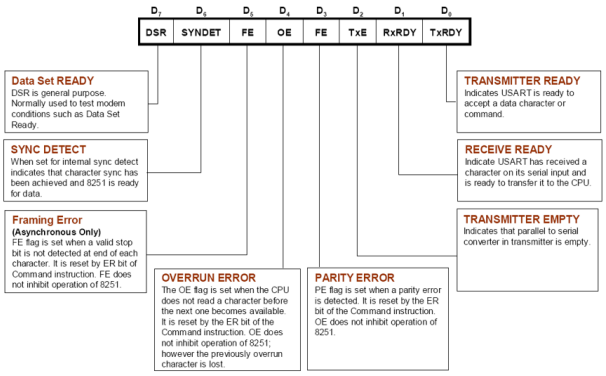
同步



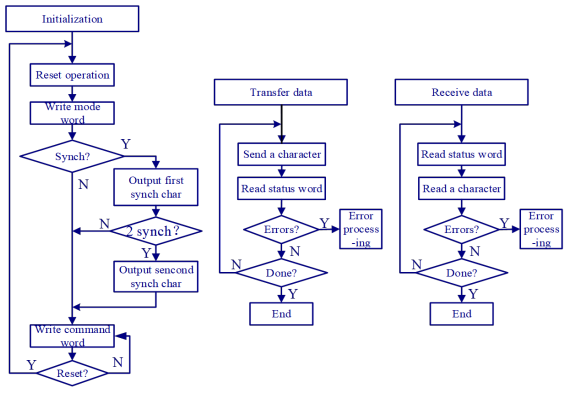
8251 命令字



8251 状态字



8251 运转



实验一
读取开关量状态取反后送显示

```
.MODEL SMALL
.DATA
.STACK 64
.CODE
PortIn EQU 90h ;定义输入端口号
PortOut EQU 0A0h ;定义输出端口号
main proc far
Again:IN AL,PortIn  ;读取开关量状态
      NOT AL        ;取反
      OUT PortOut,AL  ;送显示
      JMP Again     ;跳转循环执行
main endp
END main        ;指示汇编程序结束编译
```

样例
转换小写至大写

```
.MODEL SMALL
.STACK 64
       .DATA
DATA1  DB 'mY NAME is jOe'
       ORG 0020H
DATA2  DB  14 DUP(?)
       .CODE
MAIN   PROC FAR
       MOV AX, @DATA
       MOV DS, AX
       MOV SI, OFFSET DATA1
       MOV BX, OFFSET DATA2
       MOV CX, 14
BACK:  MOV AL, [SI]   ; get next ch
       CMP AL, 61H    ; less than 'a'
       JB  OVER
       CMP AL, 7AH    ; greater than 'z'
       JA  OVER
       AND AL, 11011111B ; convert
OVER:  MOV [BX],AL
       INC SI
       INC BX
       LOOP BACK
       MOV AH, 4CH
       INT 21H
MAIN   ENDP
       END MAIN
```

实验二
奇偶存储奇偶数

```
       .MODEL SMALL
    .8086
       .data
       .code
       .startup
       MOV AX,8000H          ;指定开始地址DS
       MOV DS,AX
       MOV BX,0H

       MOV AL,0H
L:
       MOV BYTE PTR [BX],AL  ;将中的数据以字
           节为单位送到ALDS:所指字节单元BX
```

```
        INC AL
        INC BX
        JNZ L
WT:
        JMP WT
        .stack 100h      ; 定义字节容量的堆
            栈256
        END
```

样例
输入送数码管显示

```
A_PORT EQU 8020H
B_PORT EQU 8022H
C_PORT EQU 8024H
CTRL_PORT EQU 8026H

        .DATA
TAB1    DB 3FH ; 7-Segment Tube, 0 - F
        DB 06H ; 共阴极类型的段数码管7
        DB 5BH ;    a a a
        DB 4FH ; f        b
        DB 66H ; f        b
        DB 6DH ; f        b
        DB 7DH ;    g g g
        DB 07H ; e        c
        DB 7FH ; e        c
        DB 6FH ; e        c
        DB 77H ;    d d d    h h h
        DB 7CH ; ------------------------
        DB 39H ; b7 b6 b5 b4 b3 b2 b1 b0
        DB 5EH ; DP g  f  e  d  c  b  a
        DB 79H ;
        DB 71H ;

        .CODE
    ASSUME CS:CODE, DS:DATA
START:  MOV AX, DATA
        MOV DS, AX
        MOV AL, 90H
        MOV DX, CTRL_PORT
        OUT DX, AL
ADD1:   MOV DX, A_PORT
        IN  AL, DX
        AND AL, 0FH
        MOV BX, OFFSET TAB1
        XLAT    ; is in AL
        MOV DX, B_PORT
        OUT DX, AL

        MOV CX, 0600H ;delay
ADD2:   LOOP ADD2
        JMP  ADD1
CODE    ENDS
        END START
```

实验三
8255送显示

```
; 芯片端口地址8255 (Port Address):
L8255PA     EQU 121H      ; Port A
L8255PB     EQU 123H      ; Port B
L8255PC     EQU 125H      ; Port C
L8255CS     EQU 127H      ; 8255 Control
    Register

INIT8255 PROC
        ; Init 8255 in Mode 0, L8255PA OUTPUT
            , L8255PB OUTPUT, L8255PCU OUTPUT
            , L8255PCL INPUT
        MOV AL, 10000001B
        MOV DX, L8255CS      ; should be
            assigned to DX first
        OUT DX, AL
        RET
INIT8255 ENDP
```

实验三
8253计时器

```
; 芯片端口地址8253 (Port Address):
L8253T0     EQU 100H      ; Timer0
L8253T1     EQU  102H     ; Timer1
L8253T2     EQU  104H     ; Timer2
L8253CS     EQU  106H     ; 8253 Control
    Register

INIT8253 PROC
; Set the mode and the initial count for
    Timer0
        MOV DX, L8253CS
        MOV AL, 00110110B
        OUT DX, AL      ; counter 0, mode 3,
            HEX
        MOV AX, 2710H    ; 1MHz -/10^4-> 100Hz
        MOV DX, L8253T0   ; counter 0
        OUT DX, AL
        MOV AL, AH      ; then send high byte
        OUT DX, AL
; Set the mode and the initial count for
    Timer1
        MOV DX, L8253CS
        MOV AL, 01110110B
        OUT DX, AL      ; counter 1, mode 3,
            HEX
        MOV AX, 64H    ; 100Hz -/100-> 1Hz
        MOV DX, L8253T1   ; counter 1
        OUT DX, AL
        MOV AL, AH      ; then send high byte
        OUT DX, AL
; Set the mode and the initial count for
    Timer2
        MOV DX, L8253CS
        MOV AL, 10110000B
        OUT DX, AL      ; counter 2, mode 0,
            HEX
        MOV AX, 0C350H    ; 1MHz -/5*10^4->
```

```
            20Hz
        MOV DX, L8253T2    ; counter 0
        OUT DX, AL
        MOV AL, AH      ; then send high byte
        OUT DX, A
        RET
INIT8253 ENDP
```

实验三
中断向量表(IVT)

```
        MOV  BL, IRQNum   ; BL is used as a
            parameter to call the procedure
            INT_INIT
        CALL INT_INIT   ; Procedure INT_INIT
            is used to set up the IVT
        MOV CH,0H     ; Initial counter for
            number displaying: 0001

INT_INIT PROC FAR
        CLI       ; Disable interrupt
        MOV AX, 0
        MOV ES, AX      ; To set up the
            interrupt vector table
; Put your code here
; Hint: you can use the directives such as
    SEGMENT,OFFSET to get the segment value
    and the offset of a label
        MOV BH, 0H
        MOV CL, 2H
        SHL BX, CL      ; X4
        MOV SI, BX
        MOV AX, OFFSET MYIRQ
        MOV ES:[SI],AX
        MOV AX, SEG MYIRQ
        MOV ES:[SI+2], AX
        STI
        RET       ; Do not to forget to return
             back from a procedure
INT_INIT ENDP
```

实验三
中断服务程序(ISR)

```
MYIRQ   PROC FAR
    STI
    MOV DX, L8255PC
    IN  AL, DX
    MOV BL, AL
    NOT BL
    AND BL, 80H
    OR  AL, BL          ; 0 -1-> 1
    AND AL, BL          ; 1 -0-> 0
    OUT DX, AL

    MOV BX, 0C350H
    MOV AL, BL
    MOV DX, L8253T2    ; counter 2 new count
    OUT DX, AL
    MOV AL, BH      ; then send high byte
    OUT DX, AL

    ADD CH, 1H
    CMP CH, 4H
    JAE reset
    IRET
reset:
    MOV CH, 0H
    IRET       ; Do not forget to return back
         from a ISR
MYIRQ   ENDP
```

Use 8251 to transfer 256 characters in async mode, assuming that the port addr are 208H and 209H, the baud factor is 16, and 1 stop bit, 1 start bit, no parity bit, and 8-bit character are used.

样例
8251 编程

```
        LEA DI, Buf1 ;[S]sender

        MOV DX, 209H
        MOV AL, 00H ; worst-case init
        OUT DX, AL
        CALL DELAY
        MOV AL, 00H
        OUT DX, AL
        CALL DELAY
        MOV AL, 00H
        OUT DX, AL
        CALL DELAY
        MOV AL, 40H ; reset command
        OUT DX, AL
        MOV AL, 01001110B ; mode word
        OUT DX, AL
        MOV AL, 00110111B ; command word
        OUT DX, AL
        MOV CX, 256 ; to send 256 char

; [S] sender
NEXT:   MOV DX, 209H
        IN AL, DX  ; status word
        AND AL, 01H ; TxRDY?
        JZ NEXT
        MOV AL, [DI]
        MOV DX, 208H ; data register 208H
        OUT DX, AL ; send the char
        INC DI
        LOOP NEXT

; [R] receiver
        MOV SI, 0
NEXT:   MOV DX, 209H
        IN AL, DX
        AND AL, 02H ; RxRDY?
        JZ NEXT
        MOV DX, 208H
        IN AL, DX  ; receive a char
        MOV buf2[SI], AL
        INC SI
        LOOP NEXT
```