CPU 设计: 8008

luoyin

目录

Chapter 1

微程序设计

1.1 子程序设计

- IF (Instruction Fetch): T1-T2-T3 (PCI)
- MR (Memory Read): T1-T2-T3 (PCR)
- MW (Memory Write): T1-T2-T3 (PCW)
- RR (Register Read): T4
- RW (Register Write): T5
- PCU (PC Update): T4-T5
- IOR (I/O Read): T3-T4-T5

1.2 指令组成

表 1.1: 指令组成

指令	指令码	组成
Lrr	11DDDSSS	PCO(PCL-PCH)-IF-rR-rW
LrM	11DDD111	PCO(PCL-PCH)-IF-MA(rLO-rMO)-MR-X1-rW
LMr	11111SSS	PCO(PCL-PCH)-IF-rR-MA(rLO-rMO)-MW
LrI	00DDD110	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-X1-rW
INr/DCr	00DDD00V	PCO(PCL-PCH)-IF-X1-rW
ALU OP r	10PPPSSS	PCO(PCL-PCH)-IF-rR-rW
ALU OP M	10PPP111	PCO(PCL-PCH)-IF-MA(rLO-rMO)-MR-X1-rW
ALU OP I	00PPP100	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-X1-rW
ROT	000VV010	PCO(PCL-PCH)-IF-X1-rW
JMP	01XXX100	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-PCO(PCL-PCH)-IMMa-PCU(PCHU-PCLU)
JFc/JTc	01VCC000	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-PCO(PCL-PCH)-IMMa-PCUc(PCHU-PCLU, c)
CAL	01XXX110	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-PCO(PCL-PCH)-IMMa-PCU(PCHU-PCLU)
CFc/CTc	01VCC010	PCO(PCL-PCH)-IF-PCO(PCL-PCH)-IMMb-PCO(PCL-PCH)-IMMa-PCUc(PCHU-PCLU, c)
RET	00XXX111	PCO(PCL-PCH)-IF-POP-X2
RFc/RTc	00VCC011	PCO(PCL-PCH)-IF-POPc(c)-X2
INP	0100MMM1	PCO(PCL-PCH)-IF-IO(rAO-rBO)-IOb-CO-rW
OUT	01RRMMM1	PCO(PCL-PCH)-IF-IO(rAO-rBO)-X0
HLT	0000000X	PCO(PCL-PCH)-IF
HLT	11111111	PCO(PCL-PCH)-IF

1.3 指令执行状态变化

1.4 微指令设计

1.4.1 微指令组成

- 状态码 (3 位)
- 寄存器组操作 (2 位): 输出使能, 写使能
-

1.4.2 指令分类

- $D_7D_6 = 00$: 特殊指令
  - $D_2D_1D_0 = 000$: HLT
  - $D_2D_1D_0 = 001$: HLT
  - $D_2D_1D_0 = 010$: ROT
  - $D_2D_1D_0 = 011$: RFc/RTc
  - $D_2D_1D_0 = 100$: ALU OP I
  - $D_2D_1D_0 = 101$: RST
  - $D_2D_1D_0 = 110$: LrM/LrI
  - $D_2D_1D_0 = 111$: RET
- $D_7D_6 = 01$: 跳转指令
  - $D_2D_1D_0 = 000$: JFc/JTc
  - $D_2D_1D_0 = 010$: CFc/CTc
  - $D_2D_1D_0 = 100$: JMP
  - $D_2D_1D_0 = 110$: CAL
  - $D_2D_1D_0 = XX1$: INP/OUT
- $D_7D_6 = 10$: 算术指令
- $D_7D_6 = 11$: 寄存器指令

1.4.3 微程序分类与跳转

使用 D_7D_6 进行一次分组, 使用 $D_2D_1D_0$ 进行二次分组

1.4.4 微指令转移

微指令转移按照如下计算规则:

$$A_i = \mu A_i + \sum P_i^I I_i + \sum P_i^S S_i + \sum P_i^C C_i \tag{1.1}$$

其中, μA_i 为微指令中的下一指令段, P 为微指令中的控制段, 按作用类型不同分为指令控制段 P_i^I, 状态控制段 P_i^S, 和条件控制段 P_i^C, I_i 为指令寄存器的位段, S_i 为状态寄存器的位段, C_i 为条件判定寄存器的位段.

1.4.5 微指令转移方式

- 直接转移: 微指令中的控制段均为 0, 微指令运行下一指令直接由微指令中的 μA_i 段决定.
- 按指令转移: 微指令中的指令控制段 P_i^I 不为 0, 此时, 微指令中的 μA_i 段决定跳转时的基址, $\sum P_i^I I_i$ 决定偏移量.
- 按状态转移: 微指令中的状态控制段 P_i^S 不为 0, 此时, 微指令中的 μA_i 段决定跳转时的基址, $\sum P_i^S S_i$ 决定偏移量.
- 按条件转移: 微指令中的条件控制段 P_i^C 不为 0, 此时, 微指令中的 μA_i 段决定跳转时的基址, $\sum P_i^C C_i$ 决定偏移量.
- 复合转移: 微指令中的 μA_i 段决定跳转时的基址, 结合指令控制段 P_i^I, 状态控制段 P_i^S, 和条件控制段 P_i^C 综合决定偏移量.

1.4.6 微指令转移方式

表 1.2: 微程序表

地址	微指令	状态	功能	下一微指令	下一状态	转移类型
0	PCL	T1	PCL 输出	PCH	T2	直接转移
1	PCH	T2	PCH 输出	IF, IMMa, IMMb	T3, WAIT	状态转移
2	IF	T3	DATA to IR and regB	rR, rLO, PCL, POP, POPc, rAO, X1	T4, T1, HLT	指令转移, 状态转移, 条件转移
	rR	T4	reg Read	rW, rLO	T5, T1, HLT	指令转移, 状态转移
	rW	T5	reg Write	PCL	T1, INT	指令转移, 状态转移
	rLO	T1	reg L Out	rHO	T2	直接转移
	rHO	T2	reg H Out	MR, MW	T3, WAIT	指令转移, 状态转移
	MR	T3	Memory Read	X1	T4	直接转移
	MW	T3	Memory Write	PCL	T1, INT	状态转移

1.4.7 跳转设计

1.4.7.1 IF 跳出

跳出指向

- rR: Lrr+LMr (11VVVSSS), ALU op r (10PPPSSS), 合并 (1XXXXSSS, SSS<>111)
- rLO: LrM (11DDD111, DDD<>111), ALU op M (10PPP111)
- rAO: INP+OUT (01XXXXX1)
- POP: RETURN (00XXXX11)
- PCL: JUMP, CALL (01XXXXX0)
- PCL(next): HLT, INT, NORMAL
- X1: INr/DCr

1.4.8 微指令组合逻辑

- srcM: $D_2D_1D_0$
- dstM: $D_5D_4D_3$

1.4.9 微指令表

表 1.3: 微指令表

地址	微指令	S			P			μA				
		2	1	0	2	1	0	4	3	2	1	0
00000	PCL	0	1	0	0	0	0	0	0	0	0	1
00001	PCH	1	0	0	0	0	0	x	x	x	x	x
00010	IF	0	0	1	x	x	x	0	1	x	x	x
01000	rR	1	1	1	x	x	x	x	x	x	x	x
01001	POP	1	1	1	0	0	0	x	x	x	x	x
01010	X1	1	1	1	0	0	0	x	x	x	x	x
01100	rLO	0	1	0	0	0	0	x	x	x	x	x
01101	rAO	0	1	0	0	0	0	x	x	x	x	x
01110	PCL2	0	1	0	0	0	0	x	x	x	x	x