计算机体系结构实验

实验报告　（实验一）

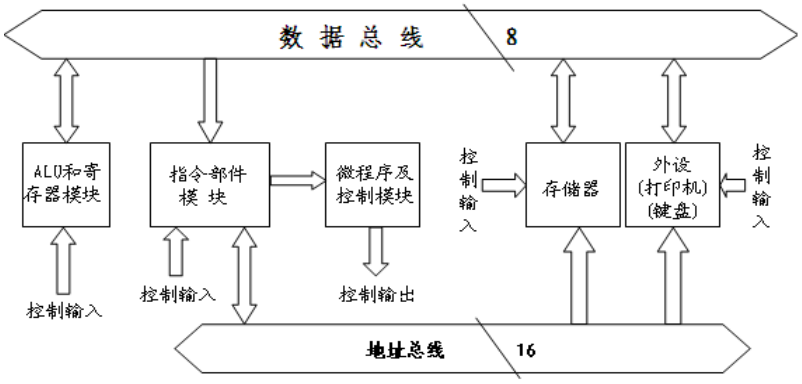
李秦琦
11300720096

目录

实验一　8 位微程序控制计算机设计

实验内容

1.1　8 位微程序控制计算机基本结构



8位单累加器计算机基本结构

总线：该计算机采用单总线，即CPU的内部总线和外部总线均为一条总线。它的结构简单，实现较方便。但它的速度较慢，无法实现流水线和并行操作。

CPU：它的CPU由ALU和寄存器模块、指令部件模块及微程序控制模块组成。

寄存器：ALU和寄存器模块包括通用寄存器（含累加器）、ALU（包括暂存器）、状态寄存器等，它可采用单累加器多寄存器或多累加器结构（见第一章）。

ALU：ALU可完成各种算术、逻辑运算功能，如加、减、与、或、取反、取负、移位等。状态寄存器可包括进位位、全零标志位、负数标志位、溢出标志位等。

指令部件：指令部件模块包括程序计数器（PC）及它的控制电路（加1、接数等）、指令寄存器（它一般包括一至三个8位寄存器，与最长的指令相对应）等。

微程序控制部件：微程序控制摸块包括微程序存贮器（现为1Kx32）、微程序计数器（MPC）及它的控制电路（加1、接数等）、微指令寄存器、时序控制电路、微操作信号发生电路等。

存贮器：存贮器为外接的RAM存贮器（现为32Kx8）

外设：包括打印机输出设备、键盘输入设备。它们均接于总线上，有分别的读、写控制信号。

1.2 指令系统和 CPU 结构

表格 1　　定义指令系统　(No. 109)

编号	汇编码	操作	指令码
1	MOV A, Ri	(Ri) → A	00000iii
2	MOV Ri, A	(A) → Ri	00001iii
3	MOV A, @Ri	(7EH[Ri]) → A	00010iii
4	MOV @Ri, A	(A) → 7EH[Ri]	00011iii
5	ADD A, Ri	(A) + (Ri) → A	00100iii
6	SUB A, Ri	(A) – (Ri) → A	00101iii
7	MOV A, #data	date → A	00110000 dddddddd
8	MOV Ri, #data	data → Ri	00111iii dddddddd
9	LDA addr	(addr) → A	01000000 addrh addrl
10	STA addr	(A) → addr	01001000 addrh addrl
11	JC addr	if CY=1 then addr→PC else (PC)+3→PC	01010000 addrh addrl
12	JMP addr	addr → PC	01011000 addrh addrl
13	JNKB addr	if KRIX=0 then addr→PC else (PC)+3→PC	01100000 addrh addrl
14	JNPB addr	if PRIX=0 then addr→PC else (PC)+3→PC	01101000 addrh addrl
15	CALL add	保存 PC 进栈，按 addr 转到子程序	01110000 addrh addrl
16	RET	PC 退栈，子程序返回	01111000
17	RSP	7fffH → SP	10000000
18	SUB A, @Ri	(A) – (7EH[Ri]) → A	10001iii
19	ASR A	(A) 算术右移一位	10010000
20	CLR addr	0 → (addr)	10011000
21	PUSH Ri	Ri → 堆栈，SP → 1	10100iii
22	POP Ri	SP+1，堆栈 → Ri	10101iii
23	JZ addr	if Z=1 then addr → PC else (PC)+3 → PC	10110000 addrh addrl
24	ADC A, Ri	if CY=1 then (A)+(Ri)+1→A else (A)+(Ri)→A	10111iii

注: 对寄存器间接寻址指令，如 MOV @Ri, A，由于 Ri 为 8 位，而存储器地址为 16 位，故取 Ri 为地址低 8 位，高 8 位固定为 7EH。

CPU 结构框图

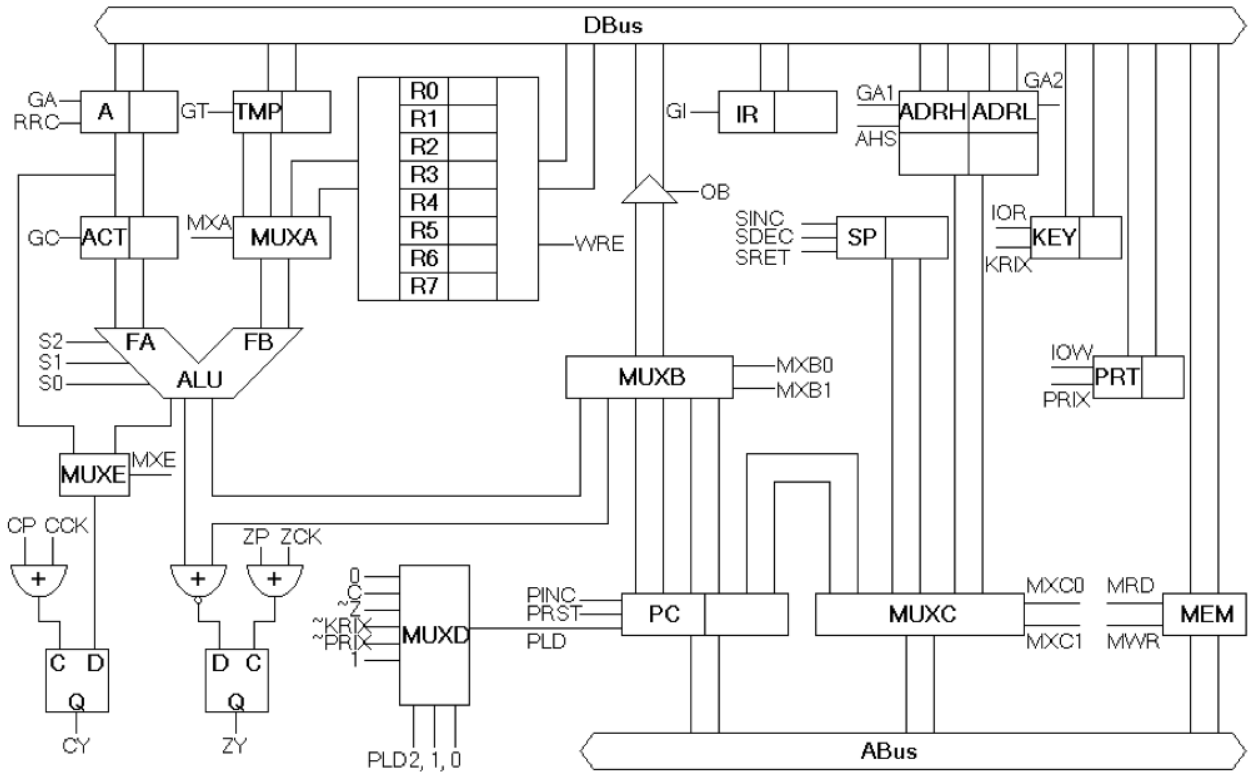


图1 CPU结构框图

1.3 设计指令流程

表格 2 指令流程定义

编号	汇编代码	指令流程
0	取指令（所有指令的最后一步）	T0: (PC) $\xrightarrow{MC=00}$ AB; (M) $\xrightarrow{CRDX=0}$ DB $\xrightarrow{GI=0}$ IR; (PC)+1 $\xrightarrow{PINC=0}$ PC; (A) $\xrightarrow{GC=0}$ ACT; MPLD=0;
1	MOV A, Ri	T1: (Ri) $\xrightarrow{MA=0\ S=011}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{GA=0}$ A;
2	MOV Ri, A	T1: (ACT) $\xrightarrow{S=010}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{WRE=0}$ Ri;
3	MOV A, @Ri	T1: (Ri) $\xrightarrow{MA=0\ S=011}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{GA2=0}$ ADRL; 7EH $\xrightarrow{AHS=0}$ ADRH; T2: (ADR) $\xrightarrow{MC=01}$ AB; (M) $\xrightarrow{CRDX=0}$ DB $\xrightarrow{GA=0}$ A;
4	MOV @Ri, A	T1: (Ri) $\xrightarrow{MA=0\ S=011}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{GA2=0}$ ADRL; 7EH $\xrightarrow{AHS=0}$ ADRH; T2: (ADR) $\xrightarrow{MC=01}$ AB; (ACT) $\xrightarrow{S=010}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{CWRX=0}$ M;
5	ADD A, Ri	T1: (ACT)+(Ri) $\xrightarrow{MA=0\ S=000}$ ALU $\xrightarrow{MB=00}$ DB $\xrightarrow{GA=0}$ A; COUT $\xrightarrow{CP=0\ ME=0}$ CY;

编号	汇编代码	指令流程
6	`SUB A, Ri`	T1: (ACT)-(Ri) $\xrightarrow{\text{MA=0 S=001}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{GA=0}}$ A; COUT $\xrightarrow{\text{CP=0 ME=0}}$ CY;
7	`MOV A, #data`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA=0}}$ A; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC;
8	`MOV Ri, #data`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{WRE=0}}$ Ri; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC;
9	`LDA addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA=0}}$ A;
10	`STA addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB; (ACT) $\xrightarrow{\text{S=010}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{CWRX=0}}$ M;
11	`JC addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=001}}$ PC;
12	`JMP addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=101}}$ PC;
13	`JNKB addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=011}}$ PC;
14	`JNPB addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=100}}$ PC;
15	`CALL add`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (SP) $\xrightarrow{\text{MC=10}}$ AB; (PCH) $\xrightarrow{\text{MB=01}}$ DB $\xrightarrow{\text{CWRX=0}}$ M; (SP)-1 $\xrightarrow{\text{SSP=01}}$ SP; T4: (SP) $\xrightarrow{\text{MC=10}}$ AB; (PCL) $\xrightarrow{\text{MB=10}}$ DB $\xrightarrow{\text{CWRX=0}}$ M; (SP)-1 $\xrightarrow{\text{SSP=01}}$ SP; T5: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=101}}$ PC;
16	`RET`	T1: (SP)+1 $\xrightarrow{\text{SSP=10}}$ SP; T2: (SP) $\xrightarrow{\text{MC=10}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (SP)+1 $\xrightarrow{\text{SSP=10}}$ SP; T3: (SP) $\xrightarrow{\text{MC=10}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; T4: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=101}}$ PC;
17	`RSP`	T1: 7FFFH $\xrightarrow{\text{SSP=11}}$ SP;

编号	汇编代码	指令流程
18	`SUB A, @Ri`	T1: (Ri) $\xrightarrow{\text{MA=0, S=011}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; 7EH $\xrightarrow{\text{AHS=0}}$ ADRH; T2: (ADR) $\xrightarrow{\text{MC=01}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GT=0}}$ TMP; T3: (PC) $\xrightarrow{\text{MC=00}}$ AB; (ACT)-(TMP) $\xrightarrow{\text{MA=1 S=001}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{GA=0}}$ A; COUT $\xrightarrow{\text{CP=0 ME=0}}$ CY;
19	`ASR A`	T1: (A) $\xrightarrow{\text{MXE=0 ASR=0}}$ CY;
20	`CLR addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB; 0 $\xrightarrow{\text{S=110}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{CWRX=0}}$ M;
21	`PUSH Ri`	T1: (SP) $\xrightarrow{\text{MC=10}}$ AB; (Ri) $\xrightarrow{\text{MA=0 S=011}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{CWRX=0}}$ M; (SP)-1 $\xrightarrow{\text{SSP=01}}$ SP;
22	`POP Ri`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (SP)+1 $\xrightarrow{\text{SSP=10}}$ SP; T2: (SP) $\xrightarrow{\text{MC=10}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{WRE=0}}$ Ri;
23	`JZ addr`	T1: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA1=0}}$ ADRH; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T2: (PC) $\xrightarrow{\text{MC=00}}$ AB; (M) $\xrightarrow{\text{CRDX=0}}$ DB $\xrightarrow{\text{GA2=0}}$ ADRL; (PC)+1 $\xrightarrow{\text{PINC=0}}$ PC; T3: (ADR) $\xrightarrow{\text{MC=01}}$ AB $\xrightarrow{\text{PLD2,1,0=110}}$ PC;
24	`ADC A, Ri`	T1: (ACT)+(Ri) $\xrightarrow{\text{MA=0 S=000}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{GA=0}}$ A; COUT $\xrightarrow{\text{CWRX=0}}$ CY; T2: (A) $\xrightarrow{\text{PLD2,1,0=001 S=111}}$ ALU $\xrightarrow{\text{MB=00}}$ DB $\xrightarrow{\text{GA=0}}$ A; COUT $\xrightarrow{\text{CWRX=0}}$ CY;

注1: 除箭头上标注的信号, 其它信号皆为无效状态。

注2：MA、MB、MC等分别为MXA、MXB、MXC的缩写。

表格 3　　微操作信号表

部件	信号	功能	有效位
累加器 **A**	GA	A 功能	GA
		累加器 A 接数允许	0
		禁止（无操作）	1
	ASR	算术右移	0
暂存器 **TMP**	GT	TMP 接数允许	0
暂存器 **ACT**	GC	暂存器接数允许	0
寄存器 **Ri**	WRE	寄存器写入允许	0
MUXA (FB 的多路选择)	MXA	ALU 的第二个操作数多路开关选择	0: 寄存器内容送 ALU 1: 暂存器 TMP 送ALU
算数逻辑单元 **ALU**	S2, S1, S0	ALU 功能	S2 S1 S0
		F = A + B	0 0 0
		F = A − B	0 0 1
		F = A	0 1 0
		F = B	0 1 1
		F = ¬B	1 0 0
		备用	1 0 1
		F = 0	1 1 0
		F = A + 1	1 1 1
CY	MXE	进位位 CY 接数 COUT	0
		CY 接数移位末位输出	1
	CP	CY 接数允许	0
指令寄存器 **IR**	GI	指令寄存器 IR 接数允许	0
地址寄存器 **ADR**	GA1	高位地址寄存器 ADRH 接数允许	0
	AHS	高位地址寄存器 ADRH 置 7EH 允许	0
	GA2	低位地址寄存器 ADRL 接数允许	0
MUXC (地址 **AB** 多路开关)	MXC1,0	MUXC 功能	MC1 MC0
		PC 送至地址总线 AB	0 0
		ADR 送至地址总线 AB	0 1
		SP 送至地址总线 AB	1 0

部件	信号	功能	有效位
		备用	1 1
程序计数器 **PC**	PINC	PC 加 1 信号	0
	PRST	PC 清 0 信号	0
MUXD (控制 **PC** 接数的多路选择**)**	PLD2,1,0	PC 接数控制	PLD2 PLD1 PLD0
		禁止	0 0 0
		CY=1 转移	0 0 1
		ZY=0	0 1 0
		KB=0	0 1 1
		PB=0	1 0 0
		必转	1 0 1
		ZY=1	1 1 0
		备用	其余
微程序计数器 **MPC**	MPLD	MPC 接数允许	0
存储器 **M**	CRDX	读存储器	0
	CWRX	写存储器	0
堆栈 **SP**	SSP	SP 减 1 信号	01
		SP 增 1 信号	10
		SP 初始化为 7FFFH	11
MUXB (数据总线 **DB** 多路开关选择**)**	MXB1,0	MUXB 功能	MXDB1 MXDB2
		ALU 结束送至数据总线 DB	0 0
		PCH 送至数据总线 DB	0 1
		PCL 送至数据总线 DB	1 0
		三态	1 1

注:

① 图1中的寄存器选择信号（RC、RB、RA）不属于微指令编码范围，应直接来自指令码（I2、I1、I0）。

② 以上的各个微操作信号，除了已标明的，可统一取 0 有效。

表格 4　　微指令控制微操作对应表

微指令的控制位	微操作	有效状态	微操作功能描述
0	WRE	0	寄存器写入允许
1	GA	0	A 功能
2	ASR	0	累加器 A 算术右移
3	GC	0	暂存器接数允许
4	GT	0	TMP 接数允许
5	GI	0	指令寄存器 IR 接数允许
6	GA1	0	高位地址寄存器 ADRH 接数允许
7	AHS	0	高位地址寄存器 ADRH 置 7EH 允许
8	GA2	0	低位地址寄存器 ADRL 接数允许
9	OB	0	MUXB 送至 DB 允许
10	MXB0	X	数据总线 DB 多路开关选择
11	MXB1	X	
12	ZP	0	ZY 接数允许
13	CP	0	CY 接数允许
14	MXE	X	进位位 CY 接数 COUT 或移位输出
15	S0	X	ALU 功能选择
16	S1	X	
17	S2	X	
18	MXA	0	ALU 的第二个操作数多路开关选择
19	PLD0	X	控制 PC 接数的多路开关选择
20	PLD1	X	
21	PLD2	X	
22	PINC	0	PC 加 1 信号
23	SSP0	X	堆栈 SP 控制信号
24	SSP1	X	
25	MXC0	X	地址 AB 多路开关选择
26	MXC1	X	
27	MPLD	0	MPC 接数允许
28	CRDX	0	读存储器 M
29	CWRX	0	写存储器 M

对应的 VHDL 硬件描述如下：

```
CWRX<=MIR(29);
CRDX<=MIR(28);
MPLD<=MIR(27);
MXC(1)<=MIR(26); MXC(0)<=MIR(25);
SSP(1)<=MIR(24); SSP(0)<=MIR(23);
PINC<=MIR(22);
PLD(2)<=MIR(21); PLD(1)<=MIR(20); PLD(0)<=MIR(19); MXA<=MIR(18);
S(2)<=MIR(17);
S(1)<=MIR(16); S(0)<=MIR(15);
MXE<=MIR(14);
CP<=MIR(13);
ZP<=MIR(12);
MXB(1)<=MIR(11); MXB(0)<=MIR(10);
OB<=MIR(9); GA2<=MIR(8); AHS<=MIR(7); GA1<=MIR(6); GI<=MIR(5);
GT<=MIR(4); GC<=MIR(3); ASR<=MIR(2); GA<=MIR(1); WRE<=MIR(0);
```

表格 5　　微指令控制微操作对应表

指令助记符	微指令	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	信号	CWRX	CRDX	MPLD	MXC1	MXC0	SSP1	SSP0	PINC	PLD2	PLD1	PLD0	MXA	S2	S1	S0	MXE	CP	ZP	MXB1	MXB0	OB	GA2	AHS	GA1	GI	GT	GC	ASR	GA	WRE
有效状态		0	0	0	X	X	X	X	0	X	X	X	0	X	X	X	X	0	0	X	X	0	0	0	0	0	0	0	0	0	0
无效状态		1	1	1	X	X	X	X	1	X	X	X	1	X	X	X	X	1	1	X	X	1	1	1	1	1	1	1	1	1	1
取指令	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	1	0	1	1	1
	001-6																														
MOV A, Ri	007	1	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1
	008	取指令																													
MOV Ri, A	00F	1	1	1	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	0
	010	取指令																													
MOV A, @Ri	017	1	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1
	018	1	0	1	0	1	0	0	1	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
	019	取指令																													
MOV @Ri, A	01F	1	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1
	020	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1
	021	取指令																													
ADD A, Ri	027	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1
	028	取指令																													
SUB A, Ri	02F	1	1	1	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1
	030	取指令																													

指令助记符	微指令	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	信号	CWRX	CRDX	MPLD	MXC1	MXC0	SSP1	SSP0	PINC	PLD2	PLD1	PLD0	MXA	S2	S1	S0	MXE	CP	ZP	MXB1	MXB0	OB	GA2	AHS	GA1	GI	GT	GC	ASR	GA	WRE
MOV A, #data	037	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
	038	取指令																													
MOV Ri, #data	03F	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	040	取指令																													
LDA addr	047	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	048	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	049	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1
	04A	取指令																													
STA addr	04F	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	050	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	051	0	1	1	0	1	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1
	052	取指令																													
JC addr	057	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	058	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	059	1	1	1	0	1	0	0	1	0	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	05A	取指令																													
JMP addr	05F	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	060	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	061	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	062	取指令																													
JNKB addr	067	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	068	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	069	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	06A	取指令																													
JNPB addr	06F	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	070	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	071	1	1	1	0	1	0	0	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	072	取指令																													
CALL	077	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	078	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	079	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	1	1	1	1	1	1

指令助记符	微指令	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	信号	CWRX	CRDX	MPLD	MXC1	MXC0	SSP1	SSP0	PINC	PLD2	PLD1	PLD0	MXA	S2	S1	S0	MXE	CP	ZP	MXB1	MXB0	OB	GA2	AHS	GA1	GI	GT	GC	ASR	GA	WRE
addr	07A	0	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	1	1	1	0	0	1	1	1	1	1	1	1	1	1
	07B	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	07C	取指令																													
RET	07F	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	080	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	081	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	082	1	1	1	0	1	0	0	1	1	0	1	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	083	取指令																													
RSP	087	1	1	1	0	0	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	088	取指令																													
SUB A, @Ri	08F	1	1	1	0	0	0	0	1	0	0	0	0	0	1	1	0	1	1	0	0	1	0	0	1	1	1	1	1	1	1
	090	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	0	1	1	1	1
	091	1	1	1	0	0	X	X	1	X	X	X	1	0	0	1	0	0	1	0	0	0	1	1	1	1	1	1	1	0	1
	092	取指令																													
ASR A	097	1	1	1	0	0	0	0	1	0	0	0	0	1	0	1	1	1	1	0	0	0	1	1	1	1	1	1	0	0	1
	098	取指令																													
CLR addr	09F	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	0A0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	0A1	0	1	1	0	1	0	0	1	0	0	0	0	1	1	0	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1
	0A2	取指令																													
PUSH Ri	0A7	0	1	1	1	0	0	1	1	0	0	0	0	0	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	1	1
	0A8	取指令																													
POP Ri	0AF	1	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0B0	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	0
	0B1	取指令																													

指令助记符	微指令	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	信号	CWRX	CRDX	MPLD	MXC1	MXC0	SSP1	SSP0	PINC	PLD2	PLD1	PLD0	MXA	S2	S1	S0	MXE	CP	ZP	MXB1	MXB0	OB	GA2	AHS	GA1	GI	GT	GC	ASR	GA	WRE
JZ addr	0B7	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	1	1	1	1	1	1
	0B8	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1	1	1	1	1	1
	0B9	1	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	0BA	取指令																													
ADC A, Ri	0BF	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1
	0C0	0	1	1	0	1	0	0	1	0	0	1	0	1	1	1	0	1	1	0	0	0	1	1	1	1	1	1	1	0	1
	0C1	取指令																													

1.4 VHDL 设计

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

-- ================================================================================
-- ================================================================================
entity cpu_8 is
    Port (
        DB:     inout std_logic_vector(7 downto 0);     --数据总线
        AB:     buffer std_logic_vector(15 downto 0);   --地址总线
        CI:     buffer std_logic_vector(31 downto 0);   --其中低10位是MPC的输出
        CO:     in std_logic_vector(31 downto 0);       --微指令输入到MIR

        --控制总线
        CLK, RESET, RUN:            in std_logic;
        MWR, MRD, IOR, IOW, MCLK:   buffer std_logic;
        --外部观察
        CTRL1, CTRL2, CTRL3, CTRL4: buffer std_logic;
        MUX:                        in std_logic_vector(2 downto 0);
        PRIX, KRIX:                 in std_logic
    );
end cpu_8;
-- ================================================================================
-- ================================================================================
architecture cpu_8_behave of cpu_8 is
--内部信号定义部分

--signal in M 存储器
    --读写控制脉冲
    signal CWR, CRD:        std_logic;
    --读写控制信号
    signal CWRX, CRDX:      std_logic;

--signal in A 累加器
    signal GA, ASR, CA:     std_logic;
```

```
    signal A:               std_logic_vector(7 downto 0);

--signal in ACT累加器暂存器
    signal ACT:             std_logic_vector(7 downto 0);
    signal GC, CC:          std_logic;

--signal in TMP 临时寄存器
    signal TMP:             std_logic_vector(7 downto 0);
    signal GT, CT:          std_logic;

--signal in Ri 寄存器
    signal R7, R6, R5, R4, R3, R2, R1, R0, ROUT:    std_logic_vector(7 downto 0);
    signal WRE, WRC:        std_logic;
    signal RS:              std_logic_vector(2 downto 0);

---多路选择器 MUXA, MUXB, MUXC, MUXD
--signal in MUXA
    signal MXA:             std_logic;

--signal in MUXB
    signal OB:              std_logic;
    signal MXB:             std_logic_vector(1 downto 0);

--signal in MUXC
    signal MXC:             std_logic_vector(1 downto 0);

--signal in MUXD
    signal PLD:             std_logic_vector(2 downto 0);
    signal PCADD:           std_logic;

--signal in MUXE
    signal MXE:             std_logic;
    signal CYA:             std_logic;
    signal CYD:             std_logic;

--signal in ALU
    signal FA, FB, FF:      std_logic_vector(8 downto 0);
    signal S:               std_logic_vector(2 downto 0);

--signal in CY
    signal CY, CP, CCK:     std_logic;

--signal in Z
    signal ZY:              std_logic;
    signal ZP:              std_logic;
    signal ZCK:             std_logic;

--signal in PC
    signal PC:              std_logic_vector(15 downto 0);
    signal PCK:             std_logic;
    signal PINC, PRST:      std_logic;
    signal ADR:             std_logic_vector(15 downto 0);      --全地址

--signal in IR
    signal IR:              std_logic_vector(7 downto 0);

    signal CCI, GI:         std_logic;

--signal in ADRH
    signal ADRH:            std_logic_vector(7 downto 0);
    signal GA1, CA1, AHS:   std_logic;

--signal in ADRL
```

```
    signal ADRL:            std_logic_vector(7 downto 0);
    signal GA2, CA2:        std_logic;

--signal in MPC
    signal MPC, MD:         std_logic_vector(9 downto 0);
    signal MCLR, MPLD:      std_logic;
    signal MPCK:            std_logic;

--signal in SP
    signal SP:              std_logic_vector(15 downto 0);
    signal SSP:             std_logic_vector(1 downto 0);
    signal SCK:             std_logic;

--signal in MIR
    signal MIR:             std_logic_vector(31 downto 0);
    signal MICK:            std_logic;

-- signal for Print and Key
    signal KOUT,POUT:       std_logic;

-- ================================================================================
-- ================================================================================

begin

--时钟信号与复位信号

---时钟信号
    pMCLK:
    process(MCLK, CLK, RESET, RUN)
    begin
    if((RUN = '0') or (RESET = '0')) then MCLK <= '0';
            elsif(CLK'event and CLK = '0') then MCLK <= not MCLK;
    end if;
    end process pMCLK;

    MPCK    <= not MCLK and CLK;
    MICK    <= not MPCK;

    WRC     <= MCLK;        --寄存器 Ri 时钟
    PCK     <= MCLK;        --PC 时钟
    CA      <= MCLK;        --A 时钟
    CT      <= MCLK;        --TMP 时钟
    CC      <= MCLK;        --ACT 时钟
    CCI     <= MCLK;        --IR 时钟
    CA1     <= MCLK;        --ADRH 时钟
    CA2     <= MCLK;        --ADRL 时钟
    CCK     <= MCLK;        --CY 时钟
    SCK     <= MCLK;        --SP 时钟
    ZCK     <= MCLK;        --ZY 时钟


----复位信号
    PRST   <= RESET;

    pMCLR:
    process(MCLK, RESET)
    begin
        if(RESET = '0') then MCLR <= '0';
            elsif(MCLK'event and MCLK = '1') then MCLR <= RUN;
        end if;
    end process pMCLR;
```

```
    CWR     <= CWRX or not MCLK;
    CRD     <= CRDX or not MCLK;

    MRD     <= CRD or AB(15);
    MWR     <= CWR or AB(15) or not CLK;
    IOW     <= not AB(15) or not AB(1) or CWR or not CLK;
    IOR     <= not AB(15) or not AB(0) or CRD;

    --功能部件

----MPC的定义
    pMPC:
    process(MPLD, MPCK, MCLR)
    begin
        if(MCLR = '0') then
            MPC <= "0000000000";
        elsif(MPCK'event and MPCK = '1') then
            if(MPLD = '0') then
                MPC <= MD;
            else MPC <= MPC + 1;
            end if;
        end if;
    end process pMPC;

    CI(9 downto 0) <= MPC;

----为程序计数器入口MD的定义
    MD(0)           <= '1';
    MD(1)           <= '1';
    MD(2)           <= '1';
    MD(7 downto 3)  <= IR(7 downto 3);
    MD(9 downto 8)  <= "00";

----MIR的定义
    pMIR:
    process(MICK)
    begin
        if(MICK'event and MICK = '1') then
            MIR <= CO;
        end if;
    end process pMIR;

----微程序控制信号
    CWRX    <= MIR(29);
    CRDX    <= MIR(28);
    MPLD    <= MIR(27);
    MXC(1)  <= MIR(26);
    MXC(0)  <= MIR(25);
    SSP(1)  <= MIR(24);
    SSP(0)  <= MIR(23);
    PINC    <= MIR(22);
    PLD(2)  <= MIR(21);
    PLD(1)  <= MIR(20);
    PLD(0)  <= MIR(19);
    MXA     <= MIR(18);
    S(2)    <= MIR(17);
    S(1)    <= MIR(16);
    S(0)    <= MIR(15);
    MXE     <= MIR(14);
    CP      <= MIR(13);
    ZP      <= MIR(12);
    MXB(1)  <= MIR(11);
    MXB(0)  <= MIR(10);
```

```
    OB      <= MIR(9);
    GA2     <= MIR(8);
    AHS     <= MIR(7);
    GA1     <= MIR(6);
    GI      <= MIR(5);
    GT      <= MIR(4);
    GC      <= MIR(3);
    ASR     <= MIR(2);
    GA      <= MIR(1);
    WRE     <= MIR(0);

----寄存器Ri的定义
    RS(2 downto 0) <= IR(2 downto 0);

    pRi:
    process(WRC, WRE, RS)
    begin
        if (WRC'event and WRC ='0') then
            if (WRE = '0') then
                case RS is
                    when "000" => R0 <= DB;
                    when "001" => R1 <= DB;
                    when "010" => R2 <= DB;
                    when "011" => R3 <= DB;
                    when "100" => R4 <= DB;
                    when "101" => R5 <= DB;
                    when "110" => R6 <= DB;
                    when "111" => R7 <= DB;
                    when others => NULL;
                end case;
            end if;
        end if;
    end process pRi;

----寄存器输出ROUT的定义
    ROUT    <=  R0 when RS = "000" else
                R1 when RS = "001" else
                R2 when RS = "010" else
                R3 when RS = "011" else
                R4 when RS = "100" else
                R5 when RS = "101" else
                R6 when RS = "110" else
                R7;

----MUXB的定义
    pMUXB:
    process(OB, MXB)
    begin
        if(OB = '0') then
            case MXB is
                when "00" => DB <= FF(7 downto 0);
                when "01" => DB <= PC(15 downto 8);
                when "10" => DB <= PC(7 downto 0);
                when others => DB <= ROUT;
            end case;
        else
            DB <="ZZZZZZZZ";
        end if;
    end process pMUXB;

----MUXC的定义
    AB <= PC                when MXC = "00" else
          ADRH&ADRL         when MXC = "01" else
```

```
          SP                  when MXC = "10";

----MUXD的定义
    PCADD   <=   '0'          when PLD = "000" else
                 CY           when PLD = "001" else
                 not ZY       when PLD = "010" else
                 not KRIX     when PLD = "011" else
                 not PRIX     when PLD = "100" else
                 '1'          when PLD = "101" else
                 ZY           when PLD = "110";

----TMP的定义
    pTMP:
    process(CT, GT)
    begin
        if(CT'event and CT = '0') then
            if(GT = '0') then
                TMP <= DB;
            end if;
        end if;
    end process pTMP;

----A的定义
    pA:
    process(CA, ASR)
    variable POOPOO:std_logic_vector(7 downto 0);
    begin
        if(CA'event and CA = '0') then
            if(ASR = '0') then
                POOPOO := A;
                CYA  <= POOPOO(0);       --移出的末位
                A    <= POOPOO(7) & POOPOO(7 downto 1);
            elsif(GA = '0') then
                A <= DB;
            end if;
        end if;
    end process pA;

----MUXE的定义
     CYD <= FF(8) when MXE = '0' else
                 CYA;

----ACT的定义
    pACT:
    process(CC, GC)
    begin
        if(CC'event and CC = '0') then
            if(GC = '0') then
                ACT <= A;
            end if;
        end if;
    end process pACT;

    FA <= '0'&ACT; --扩展成九位
    FB <= '0'&ROUT when MXA = '0' else
          '0'&TMP;

----ALU的定义
    FF  <=  FA + FB           when S = "000" else
            FA - FB           when S = "001" else
            FA                when S = "010" else
            FB                when S = "011" else
            not FB            when S = "100" else
```

```
            "000000000"     when S = "101" else
            "000000000"     when S = "110" else
            FA + '1';

----CY的定义
    pCY:
    process(CCK, CP, FF)
    begin
        if(CCK'event and CCK = '0') then
            if(CP = '0') then
                CY <= CYD;
            end if;
        end if;
    end process pCY;

----ZY的定义
    pZY:
    process(ZCK, ZP, FF)
    begin
        if(ZCK'event and ZCK = '0') then
            if(ZP = '0') then
                if(FF = "000000000") then
                    ZY <= '1';
                else
                    ZY <= '0';
                end if;
            end if;
        end if;
    end process pZY;

----PC的定义
    pPC:
    process(PCK, PRST, PCADD)
    begin
        if(PRST = '0') then
            PC <= "0000000000000000";
        elsif(PCK'event and PCK = '0') then
            if(PCADD = '1') then PC <= AB;
            elsif(PINC = '0') then
                PC <= PC + 1;
            end if;
        end if;
    end process pPC;

----IR的定义
    pIR:
    process(CCI, GI, DB)
    begin
        if(CCI'event and CCI = '0') then
            if(GI = '0') then
                IR <= DB;
            end if;
        end if;
    end process pIR;

----ADRH的定义
    pADRH:
    process(CA1, GA1, AHS, DB)
    begin
        if(CA1'event and CA1 = '0') then
            if(AHS = '0') then ADRH <= "01111110";
                elsif(GA1 = '0') then
                    ADRH <= DB;
```

```
            end if;
        end if;
    end process pADRH;

----ADRL的定义
    pADRL:
    process(CA2, GA2, DB)
    begin
        if(CA2'event and CA2 = '0') then
            if(GA2 = '0') then
                ADRL <= DB;
            end if;
        end if;
    end process pADRL;

----SP的定义
    pSP:
    process(SCK, SSP)
    begin
        if(SCK'event and SCK = '0') then
            case SSP is
                when "01" => SP <= SP - 1;
                when "10" => SP <= SP + 1;
                when "11" => SP <= "0111111111111111";
                when others => NULL;
            end case;
        end if;
    end process pSP;

----
    CI(31 downto 24)    <=  A                           when MUX = "000" else
                            PC(15 downto 8)             when MUX = "001" else
                            ADRH                        when MUX = "010" else
                            R0                          when MUX = "011" else
                            R2                          when MUX = "100" else
                            R4                          when MUX = "101" else
                            R6                          when MUX = "110" else
                            TMP;

    CI(23 downto 16)    <=  IR                          when MUX = "000" else
                            PC(7 downto 0)              when MUX = "001" else
                            ADRL                        when MUX = "010" else
                            R1                          when MUX = "011" else
                            R3                          when MUX = "100" else
                            R5                          when MUX = "101" else
                            R7                          when MUX = "110" else
                            ACT;

    CI(15 downto 12)    <=  SP(15 downto 12)            when MUX = "000" else
                            SP(11 downto 8)             when MUX = "001" else
                            SP(7 downto 4)              when MUX = "010" else
                            SP(3 downto 0)              when MUX = "011" else
                            MIR(17 downto 14);

    CI(11)              <=  KRIX                        when MUX = "000" else
                            PRIX                        when MUX = "001" else
                            OB                          when MUX = "010" else
                            MPLD                        when MUX = "011" else
                            MIR(22);

    CI(10)              <=  PRIX                        when MUX = "000" else
                            KRIX                        when MUX = "001" else
                            PINC                        when MUX = "010" else
```

```
                        GI                          when MUX = "011" else
                        CO(3)                       when MUX = "100" else
                        CO(0);

end cpu_8_behave ;
```

UCF引脚文件

```
#PINLOCK_BEGIN

#NET "DB<15>"           LOC =  "P100";
#NET "DB<14>"           LOC =  "P101";
#NET "DB<13>"           LOC =  "P102";
#NET "DB<12>"           LOC =  "P173";
#NET "DB<11>"           LOC =  "P174";
#NET "DB<10>"           LOC =  "P175";
#NET "DB<9>"            LOC =  "P176";
#NET "DB<8>"            LOC =  "P178";
NET "DB<7>"            LOC =  "P179";
NET "DB<6>"            LOC =  "P180";
NET "DB<5>"            LOC =  "P181";
NET "DB<4>"            LOC =  "P187";
NET "DB<3>"            LOC =  "P188";
NET "DB<2>"            LOC =  "P189";
NET "DB<1>"            LOC =  "P191";
NET "DB<0>"            LOC =  "P195";
NET "AB<15>"           LOC =  "P199";
NET "AB<14>"           LOC =  "P200";
NET "AB<13>"           LOC =  "P201";
NET "AB<12>"           LOC =  "P202";
NET "AB<11>"           LOC =  "P203";
NET "AB<10>"           LOC =  "P204";
NET "AB<9>"            LOC =  "P205";
NET "AB<8>"            LOC =  "P206";
NET "AB<7>"            LOC =  "P3";
NET "AB<6>"            LOC =  "P4";
NET "AB<5>"            LOC =  "P5";
NET "AB<4>"            LOC =  "P6";
NET "AB<3>"            LOC =  "P7";
NET "AB<2>"            LOC =  "P8";
NET "AB<1>"            LOC =  "P9";
NET "AB<0>"            LOC =  "P10";
NET "MUX<0>"           LOC =  "P97";
NET "MUX<1>"           LOC =  "P96";
NET "MUX<2>"           LOC =  "P95";
#NET "CO<31>"           LOC =  "P29";
#NET "CO<30>"           LOC =  "P30";
NET "CO<29>"           LOC =  "P31";
NET "CO<28>"           LOC =  "P33";
NET "CO<27>"           LOC =  "P34";
NET "CO<26>"           LOC =  "P35";
NET "CO<25>"           LOC =  "P36";
NET "CO<24>"           LOC =  "P37";
NET "CO<23>"           LOC =  "P75";
NET "CO<22>"           LOC =  "P74";
NET "CO<21>"           LOC =  "P73";
NET "CO<20>"           LOC =  "P71";
NET "CO<19>"           LOC =  "P70";
NET "CO<18>"           LOC =  "P69";
NET "CO<17>"           LOC =  "P68";
NET "CO<16>"           LOC =  "P67";
NET "CO<15>"           LOC =  "P63";
NET "CO<14>"           LOC =  "P62";
```

```
NET "CO<13>"          LOC =  "P61";
NET "CO<12>"          LOC =  "P60";
NET "CO<11>"          LOC =  "P59";
NET "CO<10>"          LOC =  "P58";
NET "CO<9>"           LOC =  "P57";
NET "CO<8>"           LOC =  "P49";
NET "CO<7>"           LOC =  "P48";
NET "CO<6>"           LOC =  "P47";
NET "CO<5>"           LOC =  "P46";
NET "CO<4>"           LOC =  "P45";
NET "CO<3>"           LOC =  "P44";
NET "CO<2>"           LOC =  "P43";
NET "CO<1>"           LOC =  "P42";
NET "CO<0>"           LOC =  "P41";

NET "CI<31>"          LOC =  "P14";
NET "CI<30>"          LOC =  "P15";
NET "CI<29>"          LOC =  "P16";
NET "CI<28>"          LOC =  "P17";
NET "CI<27>"          LOC =  "P18";
NET "CI<26>"          LOC =  "P20";
NET "CI<25>"          LOC =  "P21";
NET "CI<24>"          LOC =  "P22";
NET "CI<23>"          LOC =  "P134";
NET "CI<22>"          LOC =  "P133";
NET "CI<21>"          LOC =  "P127";
NET "CI<20>"          LOC =  "P126";
NET "CI<19>"          LOC =  "P125";
NET "CI<18>"          LOC =  "P123";
NET "CI<17>"          LOC =  "P122";
NET "CI<16>"          LOC =  "P121";
NET "CI<15>"          LOC =  "P120";
NET "CI<14>"          LOC =  "P119";
NET "CI<13>"          LOC =  "P115";
NET "CI<12>"          LOC =  "P114";
NET "CI<11>"          LOC =  "P113";
NET "CI<10>"          LOC =  "P112";
NET "CI<9>"           LOC =  "P111";
NET "CI<8>"           LOC =  "P110";
NET "CI<7>"           LOC =  "P90";
NET "CI<6>"           LOC =  "P89";
NET "CI<5>"           LOC =  "P88";
NET "CI<4>"           LOC =  "P87";
NET "CI<3>"           LOC =  "P86";
NET "CI<2>"           LOC =  "P83";
NET "CI<1>"           LOC =  "P82";
NET "CI<0>"           LOC =  "P81";
NET "MCLK"            LOC =  "P192";
NET "MRD"             LOC =  "P24";
NET "IOW"             LOC =  "P99";
NET "IOR"             LOC =  "P84";
#NET "CTRL4"           LOC =  "P168";
#NET "CTRL3"           LOC =  "P98";
#NET "CTRL2"           LOC =  "P166";
#NET "CTRL1"           LOC =  "P165";
NET "MWR"             LOC =  "P23";
NET "CLK"             LOC =  "P94";
#NET "CLKG"            LOC = "P80";
NET "RUN"             LOC =  "P109";
NET "RESET"           LOC =  "P108";
NET "PRIX"            LOC= "P193";
NET "KRIX"            LOC = "P194";
```

```
#PINLOCK_END
```

指令定义文件

```
-MOV
A,Ri
00000iii

-MOV
Ri,A
00001iii

-MOV
A,@Ri
00010iii

-MOV
@Ri,A
00011iii

-ADD
A,Ri
00100iii

-SUB
A,Ri
00101iii

-MOV
A,#data8
00110000
dddddddd

-MOV
Ri,#data8
00111iii
dddddddd

-LDA
addr
01000000
aaaaaaaa
aaaaaaaa

-STA
addr
01001000
aaaaaaaa
aaaaaaaa

-JC
addr
01010000
aaaaaaaa
aaaaaaaa

-JMP
addr
01011000
aaaaaaaa
aaaaaaaa
```

```
-JNKB
addr
01100000
aaaaaaaa
aaaaaaaa

-JNPB
addr
01101000
aaaaaaaa
aaaaaaaa

-CALL
addr
01110000
aaaaaaaa
aaaaaaaa

-RET
01111000

-RSP
10000000

-SUB
A,@Ri
10001iii

-ASR
A
10010000

-CLR
addr
10011000
aaaaaaaa
aaaaaaaa

-PUSH
Ri
10100iii

-POP
Ri
10101iii

-JZ
addr
10110000
aaaaaaaa
aaaaaaaa

-ADC
A,Ri
10111iii

-enddef
```

1.5 测试文件

1.打印两位数加法

```
1       0000        3F00        MOV R7,#0H
2       0002                        W1:
3       0002        680002      JNPB W1
4       0005        300A        MOV A,#AH
5       0007        488002      STA 8002H
6       000A                        W2:
7       000A        68000A      JNPB W2
8       000D        488002      STA 8002H
9       0010                        L2:
10      0010        600010      JNKB L2
11      0013        408001      LDA 8001H
12      0016        0A          MOV R2,A
13      0017                        L1:
14      0017        600017      JNKB L1
15      001A        408001      LDA 8001H
16      001D        09          MOV R1,A
17      001E                        WL1:
18      001E        68001E      JNPB WL1
19      0021        488002      STA 8002H
20      0024                        WL2:
21      0024        680024      JNPB WL2
22      0027        02          MOV A,R2
23      0028        3810        MOV R0,#10H
24      002A        20          ADD A,R0
25      002B        488002      STA 8002H
26      002E                        WADD:
27      002E        68002E      JNPB WADD
28      0031        3010            MOV A,#10H
29      0033        488002      STA 8002H
30      0036                        WADD1:
31      0036        680036      JNPB WADD1
32      0039        300A            MOV A,#AH
33      003B        488002      STA 8002H
34      003E                        WADD2:
35      003E        68003E      JNPB WADD2
36      0041        488002      STA 8002H
37      0044                        L4:
38      0044        600044      JNKB L4
39      0047        408001      LDA 8001H
40      004A        0C          MOV R4,A
41      004B                        L3:
42      004B        60004B      JNKB L3
43      004E        408001      LDA 8001H
44      0051        0B          MOV R3,A
45      0052                        W3:
46      0052        680052      JNPB W3
47      0055        488002      STA 8002H
48      0058                        W4:
49      0058        680058      JNPB W4
50      005B        04          MOV A,R4
51      005C        3810        MOV R0,#10H
52      005E        20          ADD A,R0
53      005F        488002      STA 8002H
54      0062                        WEQ:
55      0062        680062      JNPB WEQ
56      0065        3019        MOV A,#19H
57      0067        488002      STA 8002H
58      006A                        WEQ1:
59      006A        68006A      JNPB WEQ1
60      006D        300A        MOV A,#AH
```

```
61      006F        488002      STA 8002H
62      0072                    WEQ2:
63      0072        680072      JNPB WEQ2
64      0075        488002      STA 8002H
65      0078        04          MOV A,R4
66      0079        22          ADD A,R2
67      007A        0E          MOV R6,A
68      007B        01          MOV A,R1
69      007C        23          ADD A,R3
70      007D        0D          MOV R5,A
71      007E        380A        MOV R0, #AH
72      0080        28          SUB A,R0
73      0081        50008A      JC Z1
74      0084        0D          MOV R5,A
75      0085        3801        MOV R0,#1H
76      0087        06          MOV A,R6
77      0088        20          ADD A,R0
78      0089        0E          MOV R6,A
79      008A                    Z1:
80      008A        380A        MOV R0,#AH
81      008C        06          MOV A,R6
82      008D        28          SUB A,R0
83      008E        500094      JC Z2
84      0091        0E          MOV R6,A
85      0092        3F01        MOV R7,#1H
86      0094                    Z2:
87      0094                    W5:
88      0094        680094      JNPB W5
89      0097        05          MOV A,R5
90      0098        488003      STA 8003H
91      009B                    W6:
92      009B        68009B      JNPB W6
93      009E        06          MOV A,R6
94      009F        488003      STA 8003H
95      00A2                    W7:
96      00A2        6800A2      JNPB W7
97      00A5        07          MOV A,R7
98      00A6        3810        MOV R0,#10H
99      00A8        20          ADD A,R0
100     00A9        488003      STA 8003H
101     00AC                    LOOP:
102     00AC        5800AC      JMP LOOP
```

2.特殊指令测试

```
1       0000        3001        MOV A, #1H
2       0002        90          ASR A
3       0003        3086        MOV A, #86H
4       0005        90          ASR A
5       0006        39F1        MOV R1, #F1H
6       0008        3AF2        MOV R2, #F2H
7       000A        80          RSP
8       000B        A1          PUSH R1
9       000C        A2          PUSH R2
10      000D        3901        MOV R1, #1H
11      000F        3A02        MOV R2, #2H
12      0011        A9          POP R1
13      0012        AA          POP R2
14      0013        300F        MOV A, #FH
15      0015        3F01        MOV R7, #1H
16      0017        1F          MOV @R7, A
17      0018        987E01      CLR 7E01H
18      001B        17          MOV A, @R7
```

```
19      001C        3005        MOV A, #5H
20      001E        3D05        MOV R5, #5H
21      0020        2D          SUB A, R5
22      0021        B00026      JZ L3
23      0024        3011        MOV A, #11H
24      0026                        L3:
25      0026        30FF        MOV A, #FFH
26      0028        3900        MOV R1, #0H
27      002A        B9          ADC A, R1
```