



# 计算机体系设计实践报告

2022 学年秋季学期

序号	学号	姓名	专业	班级	成绩
	20211120138	薛凡豪	网络空间安全		
	20211120148	罗安乔	网络空间安全		

指导教师：冯立波

软件学院 计算机体系设计实践 王道

二〇二一年九月

---

## 实验 4 微程序控制器

### 实验操作

#### 1. 下载实验资源

在实验指导页面下载通用文件“DE2-115 工程模板”解压缩到 E 盘或 F 盘。将通用文件“DE2-115\_proj”解压缩到 E 盘或 F 盘。得到 DE2-115 工程文件夹。将微程序控制器中文件解压到 DE2-115 工程文件夹中；

将 lab3 中文件解压，将其中的运算器 *ALU.v*、寄存器模块 *R.v*（运算通路中的 A 暂存器和 PSW 标志寄存器由该模块实例化得到）、通用寄存器组 *GRS.v*、移位寄存器 *Shifter.v*，四个文件解压到 DE2-115 工程文件夹中。

实验电路顶层文件 *Lab\_Top.v*、微地址形成模块 *uAG.v*、时序发生器模块 *Sequencr.v*、控制存储器模块 *ControlMemory.v* 及它的 IP 核文件、寄存器模块 *R.v*（微指令寄存器 *uIR*、微地址寄存器 *uAR*、指令寄存器 *IR* 都由该模块实例化得到）。

*lab7.vpl* 和 *lab7.bmp* 是留给实验调试软件使用的虚拟面板构图文件。*Lab7\_CM.mif* 是控制存储器的初始化文件。

#### 2. 实验电路设计与下载

在工程文件夹 DE2-115 中双击工程文件 *DE2\_115\_Lab.qpf* 打开实验电路的 QuartusII 工程。

对工程进行全编译（Start Compilation）按钮，自动完成分析综合、布局布线、生成编程文件等整个过程，全编译完成后，点击工具栏中的编程按钮（Programmer），将生成的实验电路文件 *DE2\_115\_Lab.sof* 下载到实验板。

#### 3. 实验电路功能验证

打开实验调试软件 JULAB3，选择逻辑部件实验类型，在“虚拟实验板”菜单的面板构图选项下，浏览选择工程文件夹中的 *lab7.vpl* 文件，打开本实验的虚拟面板，根据实验原理，控制虚拟面板的开关、按键，观察对应的指示灯，在实验报告册中填写实验结果记录和分析。

### 实验记录

#### 1. 取指令微程序设计

取指令是任何指令执行的第一个阶段。实验电路复位时，微指令寄存器 *uIR* 清零，微地址形成模块 *uAG* 输出 00H 给控制存储器的地址，因此第一条微指令要存放在控制存储器的

00H 地址单元，即取指令微程序的入口地址从 00H 开始。

指令寄存器的内容由开关提供，因此取指令微程序只需要设计一条微指令用来产生指令寄存器的时钟使能信号 *IRce*，即微指令字段 F1 编码为 100B，同时使用固定转移方式




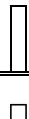

（BM=0）根据 NA 字段产生下一条微指令的微地址 01H。针对指令系统中装数和运算两类

指令，指令寄存器 IR 取到指令后，指令执行流程应根据指令操作码决定是否需要取目的操作数实现两分支转移，设计第二条微指令实现两分支转移转移（BM=1），即 F4 字段为 01B，F5 字段可以任意，考虑到地址的连续性，设置 NA 为 000010B。

指令执行阶段	微地址(H)	微指令(H)	微指令字段						微命令
			F0	F1	F2	F3	F4	F5	
取指令	00	10001	000	100	0000	00	00	000001	IRce
	01	00042	000	000	0000	00	01	000010	BM1

使用 Quartus II In-System Memory Content Editor 工具将微程序输入到控制存储器中，具体操作参阅实验指导书第五章 5.1.3 在系统存储器数据编辑器。

将指令寄存器输入端的开关 I<sub>9-6</sub> 设置为全 0，执行上面的取指令微程序，将结果填入下表；每条微指令的执行需要 2 个周期，故用两行记录。表中“有效的控制信号”一栏填写点亮的指示灯所对应的控制信号名称，如 IRce。

	RESE	Cloc	I <sub>9-0</sub>	CP1	CP2	μAR	CMdata(H)	BM	NA	有效的控制信号
			0000000000	0	1	—	—	—	—	—
①	0		—	1	0	000000	10001	0	000001	—
	0		—	0	1	—	—	—	—	IRce
②	0		—	1	0	000001	00042	1	000010	IRce
	0		—	0	1	—	—	—	—	—
③	0		—	1	0	000010	—	—	—	—
										

实验结果分析：

复位时，CP2=1，CP1=0，因此微指令执行过程中，Clock 时钟信号到来后，首先出现的是 CP1（CP1/CP2）的上升沿。

第①条微指令执行时，μAR 和控存输出 CMdata 的变化发生在 CP1（CP1/CP2）变高的时候，表明 CP1（CP1/CP2）将微地址打入 μAR，启动从控存读出微指令的操作；控制信号 IRce 的变化发生在 CP2（CP1/CP2）变高的时候，表明 CP2（CP1/CP2）将控存输出的微指令打入 μIR，开始执行这条微指令。

第②条微指令的 CP1 为 1 时 μAR=000001，表明第①条微指令的微地址转移方式为固定转移方式(BMD)，微转移地址由 NA（NA/NA 及 IR）决定。

表格第③条的设计是为了观察第②条微指令产生的微地址， CP1 为 1 时 uAR=000010，表明第②条微指令的微程序转移方式为 两分支转移，微转移地址由 NA 及 IR (NA/NA 及 IR) 决定，取到的指令是 装数指令 (装数指令/运算指令)，微程序将进入 装数指令执行阶段 (取目的操作数阶段/装数指令执行阶段)，入口地址为 000010。

将指令寄存器输入端的开关 I<sub>9-6</sub> 设置为不全 0，复位后，重新执行取指令微程序，取到的指令是 运算指令 (装数指令/运算指令)，在第③步 CP1 为 1 时 uAR= 000011，

微程序将进入 取目的操作数阶段 (取目的操作数阶段/装数指令执行阶段)，入口地址 000010。——如果想将装数指令执行阶段和取目的操作数的微指令安排在 08H~09H 地址， 01H 地址

的微指令的 NA 字段应该改成 001000。

## 2. LD R1, #0101B

### (1) 指令编码


将指令 LD R1, #0101B 翻译成二进制机器码。根据指令格式和表 3.3 指令操作码编码表，LD 指令的操作码 OPCODE (IR<sub>9</sub>~IR<sub>6</sub>) 是 0000B，INDEX (IR<sub>5</sub>~IR<sub>4</sub>) 是 01B，DATA (IR<sub>3</sub>~IR<sub>0</sub>) 是 0101B，因此翻译出指令机器码是 0000010101 B，使用开关将二进制机器码送到指令寄存器 IR 的数据输入端。

(2) 微程序设计 取指令微程序已经在前面的任务中完成，下表只包含执行和观察阶段的微指令。设计微程序并输入到控制存储器中。

指令执行阶段	微地址 (H)	微指令 (H)	微指令字段						微命令
			F0	F1	F2	F3	F4	F5	
执行	02H	64006H	11	001	0000	00	00	000110	DATA->R1
观察	06H	20000H	01	000	0000	00	00	000000	R1->BUS

(3) 微程序的执行结果记录复位后运行 LD 指令微程序，将结果填入下表。

	RESE	Cloc	CP1	CP2	μAR	CMdata	有效的控制信号	BUS	INDE
		k	0	1	—	—	—	—	X
①	0		1	0	000000	10001	—	—	—
	0		0	1	—	—	IRce	—	—
②	0		1	0	000001	00042	IRce	—	01
	0		0	1	—	—	—	—	01
③	0		1	0	000010	64006	—	—	01
	0		0	1	—	—	GRSce,DATAoe	5	01
④	0		1	0	000110	20000	GRSce,DATAoe	5	01

0		0	1	—	—	GRSoe	5	01
---	---	---	---	---	---	-------	---	----

实验结果分析：

第③条微指令的 CP2 为 1 时，INDEX=01，DATAoe=1，GRSce=1，BUS=5，也就是将 DATA 的内容送到总线上，寄存器 R1 将在 CP2（CP1/CP2）变高的时候，保存总线上的内容。

### 3. 改变 LD 指令操作码

将 LD 指令的操作码 OPCODE 改为 1111B，需要将 uAG.v 代码的第 4 行修改为 2'b01:uAGOut=SNA[5:1],IR[9:6]。

完成代码修改后，重新编译 QuartusII 工程并下载，试一试修改后的 LD 指令在取指令结束后能否转移到 02H 微地址正确运行。

### 4. ADD R1, #0111B

#### (1) 指令编码

将指令 ADD R1, #0111B 翻译成二进制机器码。根据指令格式和表 3.3 指令操作码编码表，ADD 指令的操作码 OPCODE（IR<sub>9</sub>~IR<sub>6</sub>）是 0001，INDEX（IR<sub>5</sub>~IR<sub>4</sub>）是 01，DATA（IR<sub>3</sub>~IR<sub>0</sub>）是 0101B，因此翻译出指令机器码是 0001010101B，使用开关将二进制机器码送到指令寄存器 IR 的数据输入端。

#### (2) 取目的操作数的微程序设计

取目的操作数指将寄存器 Ri 的值取出后保存在 A 中，由指令的 INDEX 字段指定寄存器，因此设计一条微指令产生控制信号 GRSoe 和 Ace。考虑到地址的连续性，下一条微指令的微地址设计为 04H，即设置 BM 为 00B，NA 为 000100B。

目的操作数取到以后，需要根据指令操作码生成各条运算类指令执行阶段的微程序入口地址，因此接下来设计的一条微指令，用于实现多分支转移（BM=2），即 F4=10，F5 字段在多分支转移方式下不影响微地址生成，可以为任意值。在 uAG 模块中实现 BM=2 的代码是 2'b10:uAGout={2'b01,IR[9:6]};，据此可计算出各运算类指令执行阶段的微程序地址范围是 11H~1CH。
















































































指令执行阶段	微地址(H)	微指令(H)	微指令字段						微命令
			F0	F1	F2	F3	F4	F5	
取目的操作数	03H	28004	01	010	0000	00	00	000100	R1->A
	04H	0008B	00	000	0000	00	10	000101	BM2

#### (3) ADD 指令执行阶段的微程序设计

指令执行阶段	微地址(H)	微指令(H)	微指令字段						微命令
			F0	F1	F2	F3	F4	F5	
执行	11H	6C705	11	011	0001	11	00	000101	ADD
保存	05H	44006	10	001	0000	00	00	000110	SHIFTER->R1
观察	06H	20000	01	000	0000	00	00	000000	R1->BUS

取指令和取目的操作数的微程序在前面的任务中已经输入控制存储器，继续使用 Quartus II In-System Memory Content Editor 工具将后续执行等阶段的微程序输入到控制存储器中。

#### (4) 微程序的执行结果记录

	RESET	Clock	$\mu$ AR	CMdata	有效的控制信号	A	BUS	S_Q	PSW	INDE
										X 
①	0		000000	10001						
	0				IRce					
②	0		000001	00042	IRce					01
	0									01
③	0		000011	28004						01
	0				GRSoe,Ace		5			01
④	0		000100	0008B	GRSoe,Ace	5				01
	0					5				01
⑤	0		001011	6C705		5				01
	0				DATAoe,PSWce	5	5			01
⑥	0		000101	44006	DATAoe,PSWce	5	5	0	0000	01
	0				GRSce,Soe	5	C(H)	C(H)	1000	01
⑦	0		000110	20000	GRSce,Soe	5	C(H)	C(H)	1000	01
	0				GRSoe	5	C(H)	0(H)	0000	01

#### 实验结果分析：

前面任务的 LD 指令完成后，R1 寄存器中的值为 0101B，微程序执行完后，R1 寄存器中的值应该是 AH。反复调试执行 ADD 指令的过程中，可能使 R1 寄存器的值发生变化，观察加法指令结果时注意以当次执行过程中从 R1 寄存器取到 A 寄存器中的值为准。

第③条微指令将 R1 的内容送到 A 暂存器，但是 A 暂存器内容的变化发生在  $\mu$ AR= 时的 CP1 (CP1/CP2) 上升沿，说明 下一条 (当前 / 下一条) 微指令地址打入  $\mu$ AR 的同时，当前 (当前 / 下一条) 微指令的执行结果打入寄存器保存。

PSW 和 SHIFTER 的变化发生在 CP2 (CP1/CP2) 变高的时候，表明 CP1 (CP1/CP2) 将微指令的执行结果打入运算器数据通路中的寄存器保存。

和实验 3.3 手动产生控制信号相比，用微指令产生控制信号更要注意时序，哪些信号应该在一条微指令中产生、哪些信号不能同时产生。从上面的实验可以看出，完成一次 ALU 运算需要 个步骤。

仿照上述步骤，验证其它运算类指令。

## 5. 修改微地址分配

将 uAG 代码的第 13 行修改为  $uAGOut = \{2'b10, IR[9:6]\}$ ，使运算类指令执行阶段的微程序安排在 21H~2FH。完成代码修改后，重新编译 QuartusII 工程并下载；设置指令寄存器 IR 的输入  $I_{9-6}$  为 0001，复位后重新执行，取目的操作数完成后，微程序转移到地址 21H。

## 6. 修改指令系统（选做）

增加寻址方式，使得源操作数不仅可以来自于立即数，也可以来自于寄存器，例如可以实现指令：

ADD R1, R2 修改指令格式如下：

10	7	6	5	4	3	0
OPCODE		INDEX		M	DATA / INDEX2	

其中 M 用于表明源操作数是来自寄存器（由  $IR_{1-0}$  指定）还是立即数。提示：需要修改实验电路硬件，如修改 IR 寄存器，修改 uAG 代码以增加依据源操作数的两分支微转移方式，增加微命令选择寄存器号来自于 INDEX 或 INDEX2。

重画指令执行流程图，把取立即数从执行阶段分离出来，增加取源操作数阶段。设计微程序，运行微程序，记录执行结果。

# 实验小结及实验分工

## 一、收获和心得

1. 通过本次实验，了解了微程序控制器的组成主要有控制存储器、微指令寄存器和地址转移逻辑三个部分
2. 理解了微程序控制器的控制时序以及微程序控制信号的产生原理
3. 熟练掌握了微程序设计的方法以及微地址形成的方法