简易计算机的设计与实现

西安交通大学

软件学院

数字逻辑综合实验

2017 年 6 月 20 日

组　　长 易　凯 2151601053

成　　员 庞建业 2151601012

屈　松 2151601067

邢贺宇 2151601070

肖　飞 2150900084

联系邮箱 williamyi96@gmail.com

联系电话 13772103675

项目网站 https://github.com/WilliamYi96/SimpleComputer

提交日期 2017 年 6 月 20 日

目录

1 致谢

Give my sincere thanks to Teacher Zhang for his excellent teaching skills and serious altitude for our homework. Give my sincere thanks to some students who have helped us and who have inspired us when discussing with them. Give my sincere thanks to ourselves because we have overcome all difficulties and have successfully finished our digital logic assignments – simple computer. Give my sincere thanks to those pioneers who have devoted themselves to writing immortal books. Give my sincere thanks to all the people sharing their ideas and harvests without pay in QA communities.

衷心感谢张琴老师出色的教学风范以及严谨的治学态度，衷心感谢那些在讨论中帮助和启发我们的同学们，衷心感谢克服了种种困难最终完成了数字逻辑综合实验的我们，衷心感谢那些写了不朽著作为后人指明道路的先驱们，衷心感谢那些在问答社区无私奉献自己智慧成果的所有同仁！

2 关于本报告的说明

本实验报告是西安交通大学 2015 级软件工程系本科生以《数字逻辑专题实验》设计为契机进行的《简易计算机的设计与实现》。小组共有 5 人，分别为软件 51 班庞建业，软件 53 班易凯，软件 54 班屈松、邢贺宇和肖飞。小组成员之间分工明确，严格采取工程模块化的设计思想，务必追求严禁，考虑周全。经过接近两个月的设计与实现，最终于 2017 年 6 月末完成整体设计，并通过 Quartus 仿真以及验收。

组长易凯主要负责总体设计，完成的部分主要有指令集设计，T 计数器，RAM 逻辑结构设计与测试样例设计，各模块集成，实验报告编写；庞建业主要完成 8 位两路、三路、四路、五路选择器的设计实现与调试；屈松主要负责程序计数器，指令寄存器，逻辑处理单元的设计与实现；邢贺宇主要完成的是单时钟、双时钟 RAM 的设计与实现；肖飞主要完成的是计算机的仿真、下载以及逻辑设计的部分 bug。同时，对于他人的部分彼此之间有相互的协作。

此实验报告为五位成员集体的劳动成果，读者可以下载学习共同探讨心得，相关程序已经发布在 GitHub 上，

项目地址为 https://github.com/WilliamYi96/SimpleComputer.，欢迎读者提出宝贵意见。

3 实验研究基本背景

3.1 简介

计算机作为当代社会最典型、最常用，同时也是最为复杂的数字系统，在各行各业中越来越发挥着不可替代的作用。虽然当下以布尔代数为基础的计算机系统发展相对成熟，但是仍然具有着极高的学习和研究意义。

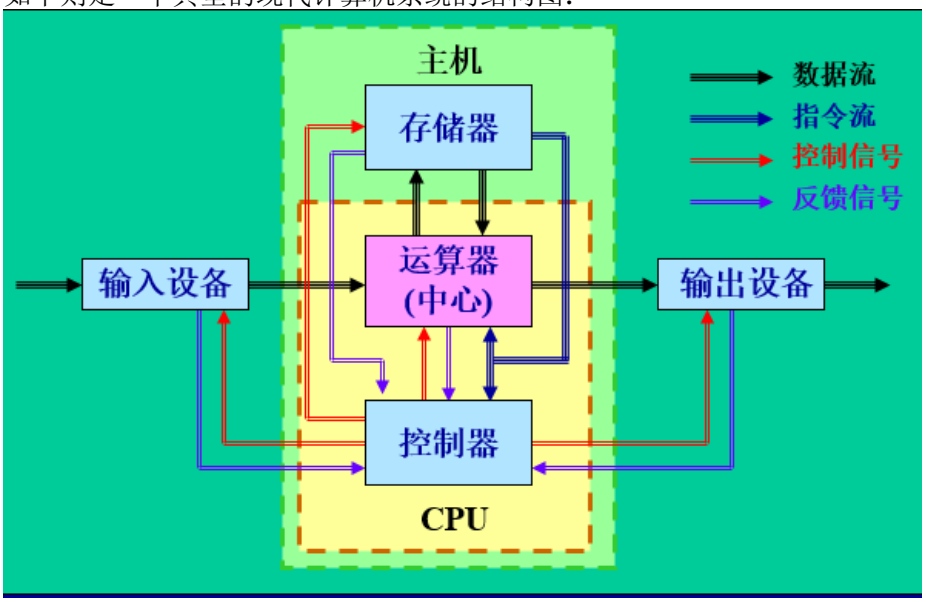
3.2 计算机体系结构

现代计算机的结构遵循的是冯·诺依曼计算机的定义，也就是说计算机由五大部件构成，其分别为运算器 ALU(算法逻辑部件)、控制器 CU、存储器 RAM(随机存取存储器) 和 IO(输入输出设备)。

由于运算器和控制器在逻辑关系和电路结构上联系的密切性，尤其是在大规模集成电路制作工艺出现之后，往往将运算器和控制器制作在同一芯片上，称为中央处理器，简称为 CPU。

现代计算机可认为是由 CPU、存储器和 I/O 设备这三个部分构成。

如下则是一个典型的现代计算机系统的结构图：



3.3 计算机各部件基本功能

CPU 的基本功能：

根据程序执行所期望的信息处理，其中 ALU 执行算术逻辑运算，CU 根据程序从存储器中读出并执行指令和数据，以执行要求的运算和操作。

存储器的基本功能：

计算机存储指令、数据和信息的空间；

输入输出设备的基本功能：

输入设备将指令、数据和信息输入到计算机，输出设备将处理后的结构由计算机输出出来。

3.4 计算机的指令系统

指令就是用以表示计算机微操作序列的二进制代码，它决定了 CPU 应该执行什么样的具体操作。

操作系统就是一台计算机所能够执行的全部指令，在某种程度上，指令系统的完备与否以及功能强弱体现了计算机整体性能的优劣。同时，计算机设计中最基本的问题就是如何选择和设计出一个完备的、使用方便的指令系统。

由于我们的出发点是在学习的基础之上加深对计算机的认识，以制作模型机为目标，以期该计算机能够满足基本的运算操作，因此我们实际设计时不考虑指令系统的完备性，重点放在考察该系统的可用性。

3.5 计算机的工作原理

计算机的指令和数据都存储在存储器中，控制器每次从存储器中取出一条指令，它决定 CPU 应该执行什么样的操作具体操作，解释其含义，并据此产生一系列功能来执行此指令。

处理一条指令所包含的操作序列称之为一个指令周期，其中，

$$\text{指令周期} = \text{取值周期} + \text{执行周期}。$$

取指周期指的是将指令从存储器中读出的操作序列，执行周期由三部分构成，分别为指令译码、取操作数、完成操作。

另外，值得注意的是，不同指令有不同字长、寻址方式、不同的操作，则其所对应的指令周期也长短不同。

同时，每个指令周期划分为若干个机器周期，每个机器周期划分为若干个节拍，一个节拍通常对应一个时钟周期 T，时钟周期是执行微操作的最小时间单位。

3.6 计算机的设计过程

计算机的设计可以分为两个阶段，分别为系统设计阶段和逻辑设计阶段。

系统设计： 设计系统的技术指标以及总的性能，确定设计目标、基本结构方案和指令系统。

逻辑设计： 将计算机结构的描述用逻辑电路来实现。

4 实验目的

我们希望通过我们数字逻辑的所学，以及查找相关的资料，按照典型计算机的实现原理来构建简易型的计算机，以满足基本的数据处理需要。

同时，希望在学习实践制作中级计算机的过程中掌握数字逻辑设计的基本方法以及重要思想，在广泛查阅相关资料的基础之上对于现代化的许多硬件设计以及计算机设计方法有一个基本的了解，另外加深对于计算机体系结构的认识，为以后的学习研究打下坚实的基础。

5 实验原理

依据计算机设计实现的框图，该计算机设计与实现的基本原理大致步骤为：

1. CLK 脉冲进行数据输入，时钟计数器 T-COUNTER 产生时钟信号 t；
2. 设置 PC 指向 00000000 地址处，从此处开始取值；
3. 将取得地址的低四位传递给 IR 进行译码，读出对应的执行指令；
4. 根据 logicProcess 的组合逻辑，结合 ti 与 qi 算出对应的 xi；
5. 根据计算得到的 xi 进行相应的逻辑操作。

6 系统设计

综合考虑实际需要，该中级计算机系统将包括 3 个寄存器, 分别为 A，B 和 R。另外有多个译码器等 (进行了器件的封装)。

6.1 存储器

每个存储器都是 256 个字，其中每个字 8 位；
该存储器具有以下寄存器：

存储地址寄存器 MAR： 与存储器的地址总线相连，存放着将要访问的存储单元的地址；

存储缓冲寄存器 MBR： 与存储器的数据总线相连，存放着要写入或者刚从存储单元读出的信息。

如果读出的为：**指令 = 操作码部分 + 地址部分**，则将操作码部分送至指令寄存器 IR，将保留在 MBR 中的地址部分发送至 MAR。

如果读出的为：**操作数**，则将其放在寄存器 A 和 R 中.

程序计数器 PC： 存放的是后续指令的地址，且有计数功能。

指令寄存器 IR： 存放的是现行指令的操作码。

操作译码器： 对指令寄存器 IR 提供的每个操作码译码出一个相应的输出变量 q_i。

时序译码器： 提供计算机时序信号 t1 到 t30 这 8 个时钟信号。

时序计数器： 为计算机提供时序信号的计数器。

以下为寄存器、存储器及其对应功能图表表示：

6.2 寄存器，存储器及其功能

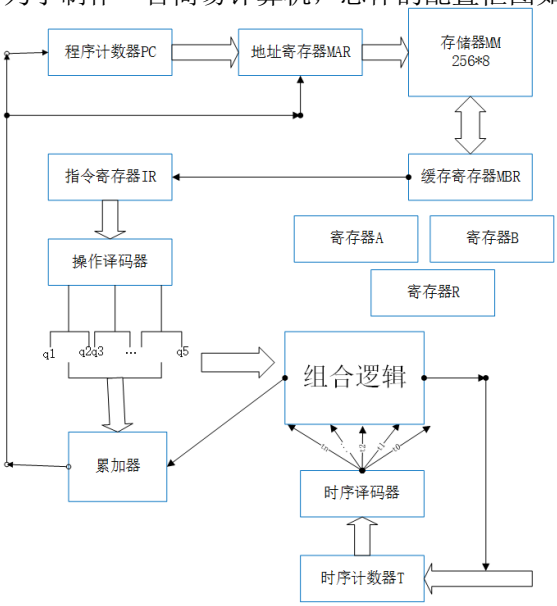
符号	位数	名称	功能
MAR	8	存储地址缓存器	保存存储器地址
MBR	8	存储缓冲寄存器	保存存储器内容
A	8	寄存器 A	处理数据寄存器
B	8	寄存器 B	处理数据寄存器
R	8	寄存器 R	处理数据寄存器
PC	8	程序计数器	保存指令地址
IR	8	指令寄存器	保存指令操作码
T	8	时序计数器	产生时序信号

表 1: 寄存器及其功能

7 逻辑设计

7.1 基本配置框图

为了制作一台简易计算机，总体的配置框图如下所示：



7.2 时钟序列设计与实现

基本功能

时钟的基本功能是产生 t0 29 的时间控制序列，其由两个 MSI 计数器 (74163) 构成，其中由于仅有两输入，五输出，根据实际需要，仅仅使用级联全加器的低五位。

单元输入与输出

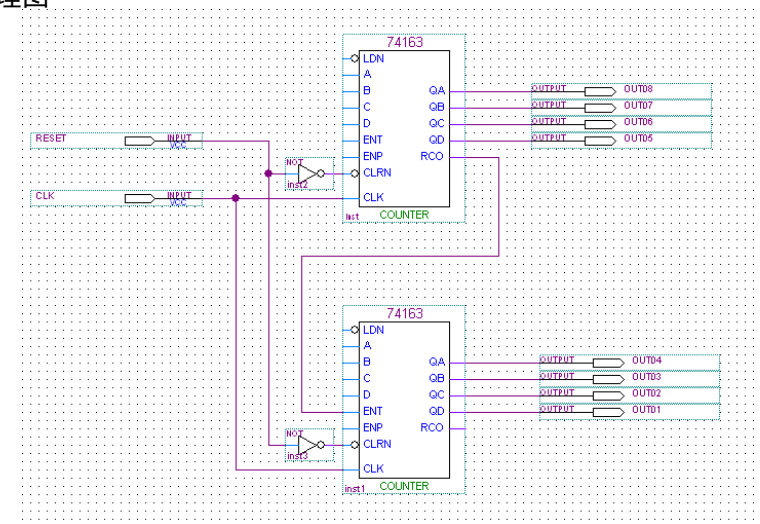
单元输入

1. CLK – 时钟，上升沿触发使计数 +1；
2. RESET – 当一条指令执行完毕之后进行 t 重置

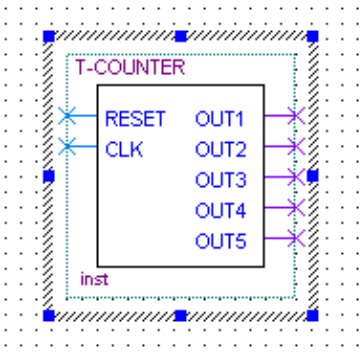
单元输出

1. OUT1 – 五位计数器的最高位
2. OUT2 – 五位计数器的第二高位
3. OUT3 – 五位计数器的第三高位
4. OUT4 – 五位计数器的第二低位
5. OUT5 – 五位计数器的最低位

原理图



封装逻辑图



7.3 程序计数器 PC 设计与实现

基本功能

程序计数器 PC 是由 8 个 D 触发器构成的模 256 计数器，其实现的是进行 RAM 中地址指针的功能，根据逻辑需要可以进行 +1 操作。

单元输入与输出

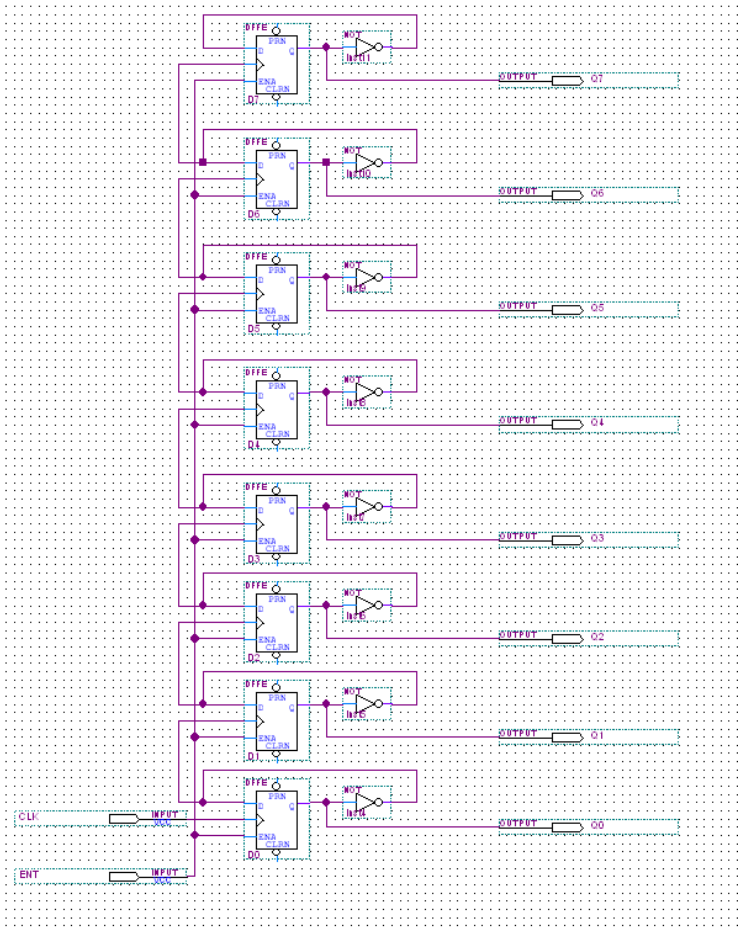
单元输入

1. CLK – 时钟脉冲，当上升沿触发 PC 计数 +1

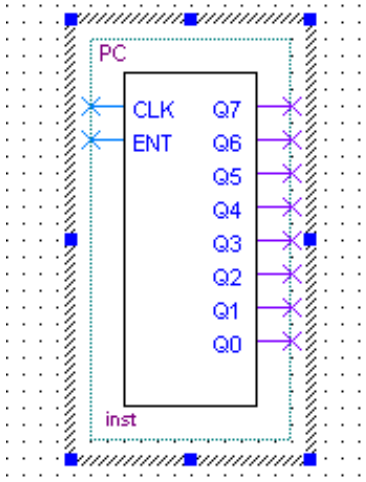
单元输出

1-7. Q0 Q7 依次为 PC 指向地址的对应位的值 (其中 Q0–Q7 升序排列)

原理图


DFFE
PRN
D
Q
ENA
CLRN
Q7
Q6
Q5
Q4
Q3
Q2
Q1
Q0
NOT
OUTPUT
CLK
ENT
INPUT
VCC

封装逻辑图



MAR, MBR 设计与实现

基本功能

MAR 是存储地址寄存器，MBR 是存储数据寄存器。其中两种均通过 74377 进行实现，由于设计较为直接，因此没有进行封装。

单元输入与输出

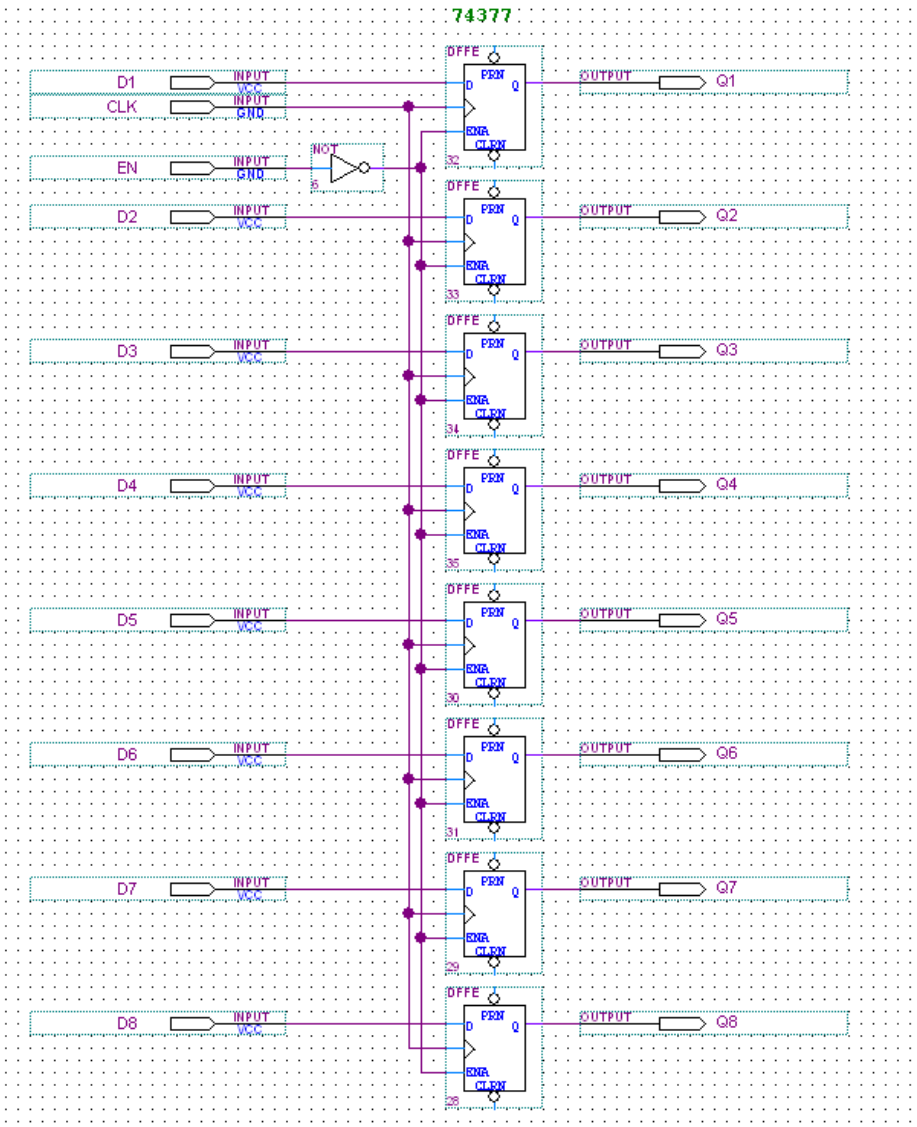
单元输入

1. CLK – 脉冲时钟，上升沿触发
2. /EN – 低使能有效
3. D1 D8 – 8 位待存储数据地址

单元输出

1. Q1 Q8 – 8 为存储数据地址输出

原理图



7.4 指令寄存器 IR

基本功能

IR 实现的功能是将从 MBR 得到的指令低四位进行译码 (由于设计指令集的原因，10 多条指令中只使用第四位就可以实现功能性译码)。然后将译码得到的值传送给 logicPr2。

单元输入与输出

单元输入

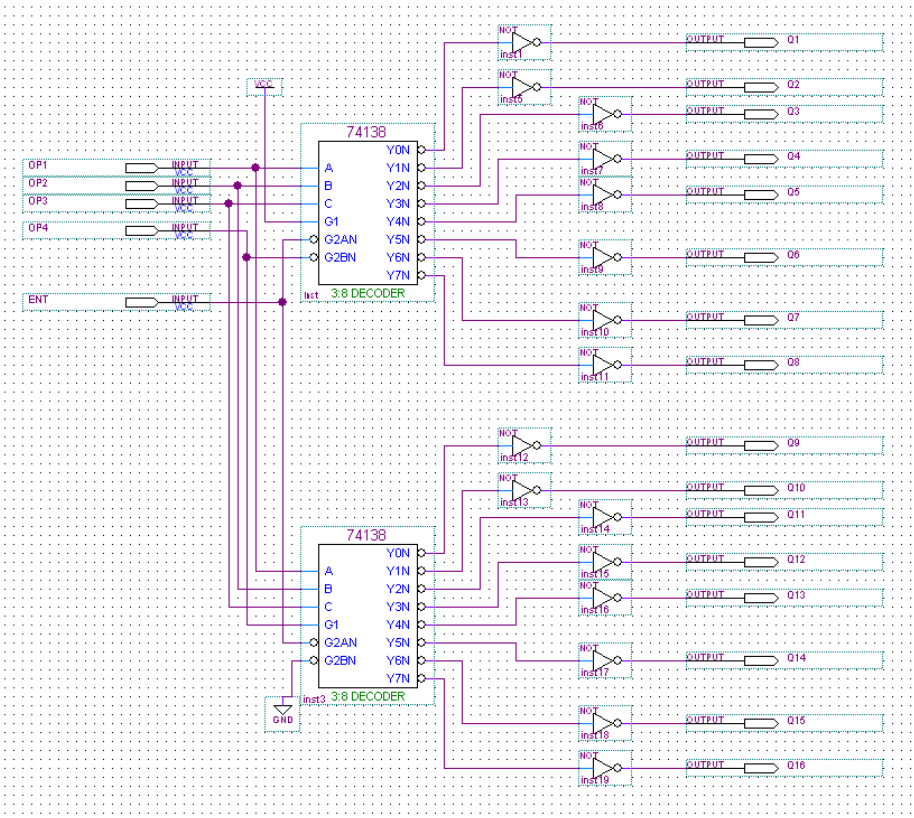
1. OP1 OP4 – 为指令的第四位 (OP1–OP4 按地址位升序排列)
2. EN – IR 指令译码使能控制

单元输出

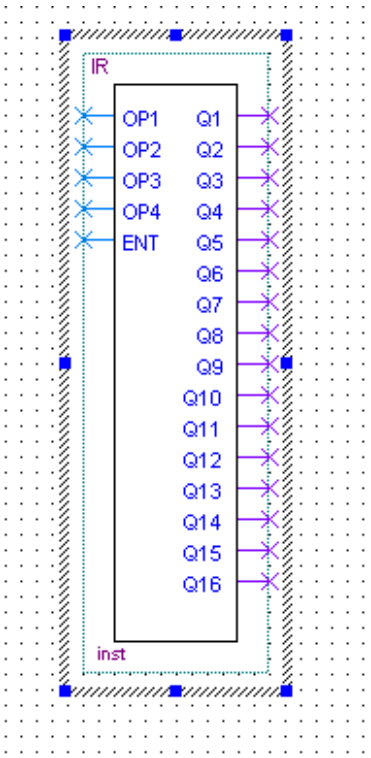
输出为对应的指令，其中对应关系如下表所示：

操作码	助记符	qi	执行功能
0 0 0 0 0 0 0 0	LDA addr	q1	将地址 addr 中的内容存入 A 寄存器处
0 0 0 0 0 0 0 1	STA addr	q2	将 A 寄存器中的内容存入地址 addr
0 0 0 0 0 0 1 0	ADD addr	q3	将地址 addr 的内容加上 A 寄存器内容，存入 A 寄存器
0 0 0 0 0 0 1 1	SUB addr	q4	将 A 中内容减去 addr 内容，并将结果存入 A 寄存器
0 0 0 0 0 1 0 0	AND addr	q5	将 addr 中内容与 A 寄存器中的内容进行与操作
0 0 0 0 0 1 0 1	OR addr	q6	将 addr 中的内容与 A 寄存器中的内容进行或操作
0 0 0 0 0 1 1 0	MOV A, R	q7	将 R 中的内容存入 A 寄存器
0 0 0 0 0 1 1 1	MOC B, R	q8	将 R 中的内容存入 B 寄存器
0 0 0 0 1 0 0 0	LDI A, OPRD	q9	将操作数存入 A 寄存器
0 0 0 0 1 0 0 1	LDI B, OPRD	q10	将操作数存入 B 寄存器

表 2: 简易计算机指令集


OP1
OP2
OP3
OP4
ENT
INPUT
VCC
74138
A
B
C
G1
G2AN
G2BN
Y0N
Y1N
Y2N
Y3N
Y4N
Y5N
Y6N
Y7N
inst 3:8 DECODER
inst3 3:8 DECODER
GND
NOT
OUTPUT
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
Q9
Q10
Q11
Q12
Q13
Q14
Q15
Q16

封装逻辑图



7.5 寄存器 A,B,R 设计与实现

基本功能 分别实现三个数据寄存器的存储功能，由于 A,B,R 逻辑结构基本相同，因此综合进行分析。

单元输入与输出

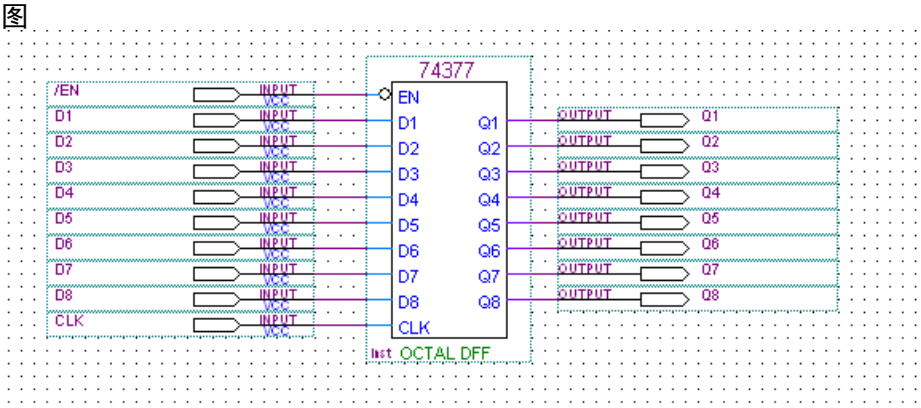
单元输入

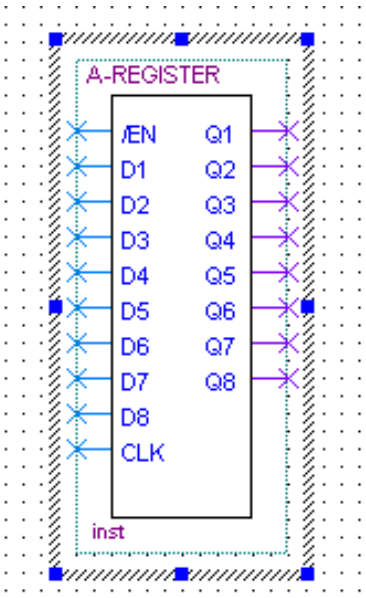
1. CLK – 脉冲时钟，上升沿触发
2. /EN – 低使能有效
3. D1 D8 – 8 位待存储数据地址

单元输出

1. Q1 Q8 – 8 为存储数据地址输出

原理图





封装逻辑图

7.6 指令译码电路的设计与实现

基本功能

实现组合逻辑 xi = qiti 的输出

单元输入与输出

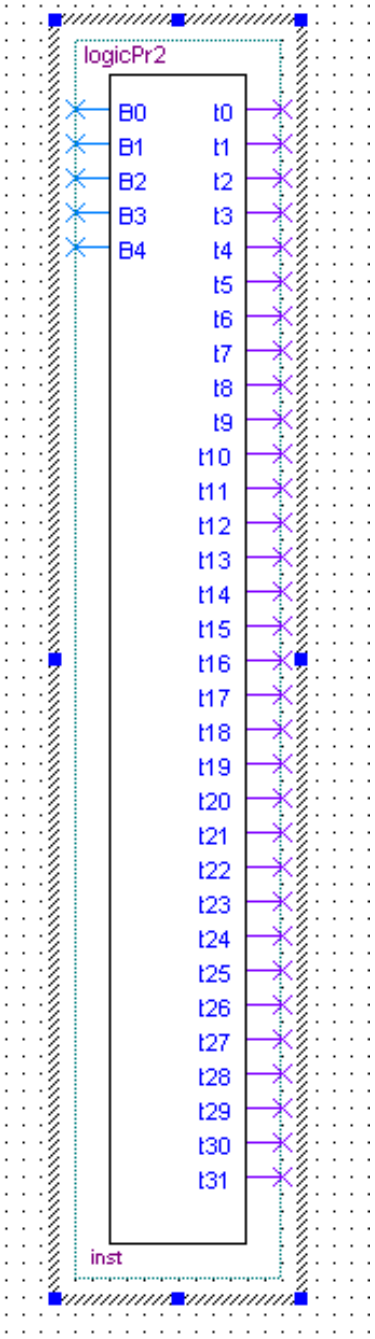
单元输入 1. OP1 OP4 – 待译码指令低四位

2. B0 B4 – 时钟序列 t 的二进制表示形式

3. ENT – 高使能

单元输出 1. x1 x29 根据逻辑实现的微操作序列

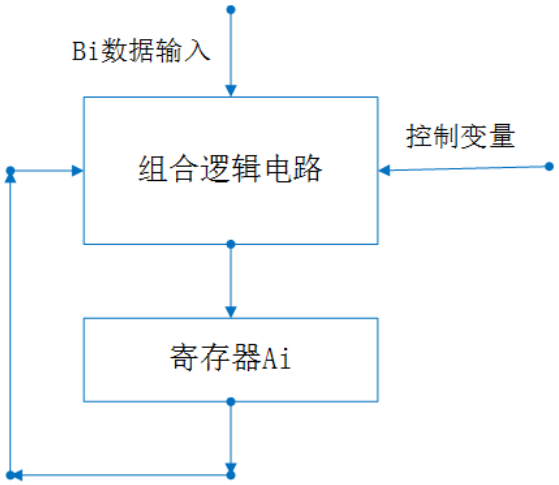
原理图 由于原理图过大过于复杂，因此不在此处展示，详情参见代码。


logicPr2
B0
B1
B2
B3
B4
t0
t1
t2
t3
t4
t5
t6
t7
t8
t9
t10
t11
t12
t13
t14
t15
t16
t17
t18
t19
t20
t21
t22
t23
t24
t25
t26
t27
t28
t29
t30
t31
inst

封装逻辑图

7.7 运算累加器设计与实现

我们累加器的实现使用 8 位 JK 触发器与组合逻辑电路进行实现，对于任何一个逻辑单元，具有如下的分析框图：



寄存器 A 既可以作为加数计数器，又可以作为和数计数器。决定累加器微操作的各个控制变量 Pi 是互斥的，在任何给定的时间内只有一个控制变量 Pi 被选通，产生相应的微操作。

为简化累加器的设计，我们假设累加器由 8 个相同的单元组成，每个单元包含了执行各种微操作所需的逻辑电路，只要完成一个单元的各部分电路设计，就可以将他们综合成了累加器的一个典型单元，然后用若干个典型单元组成一个完整的累加器 (迭代设计)。

因此我们可以通过迭代的方式设置 8 个完整的逻辑单元。

7.8 RAM 设计与实现

基本功能

存储数据及其地址的映射表。

单元输入与输出

单元输入

1. DATA1 DATA8 – 写使能时读入的数据
2. ADDR1 ADDR8 – 传入的数据或指令地址

3. WREN – 读写使能控制，当 WREN=1 时，写使能，当 WREN=0 时，读使能；

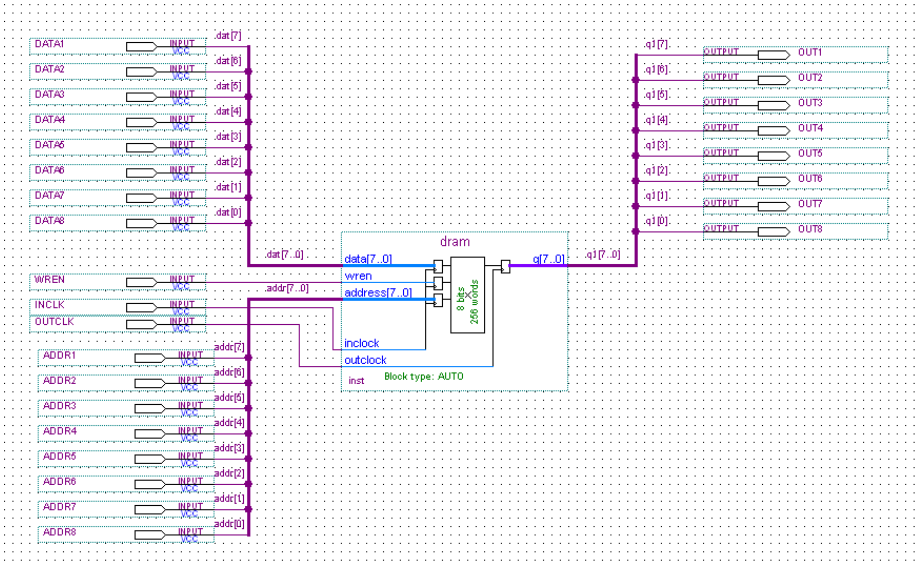
4. INCLK – 输入时钟控制

5. OUTCLK – 输出时钟控制

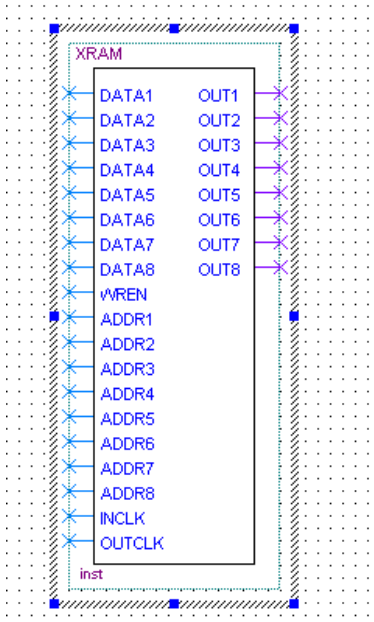
单元输出

OUT1 OUT8 – 指令或数据输出

原理图



封装逻辑图


XRAM
DATA1
DATA2
DATA3
DATA4
DATA5
DATA6
DATA7
DATA8
WREN
ADDR1
ADDR2
ADDR3
ADDR4
ADDR5
ADDR6
ADDR7
ADDR8
INCLK
OUTCLK
OUT1
OUT2
OUT3
OUT4
OUT5
OUT6
OUT7
OUT8
inst

8 实验平台

8.1 实验设备

Cyclone II EP2C8T144C8 试验箱.

8.2 实验器件

相关数据转换线，Win XP 系统台式电脑, Ubuntu gnome 16.01 之下 Win 7 虚拟机.

8.3 软件工具

QUARTER II 9.0, 截图工具，win 系统画图软件，Latex texlive，Win Edt 10.2, Sublime text 3, MiKTeX.

9 实验步骤及过程

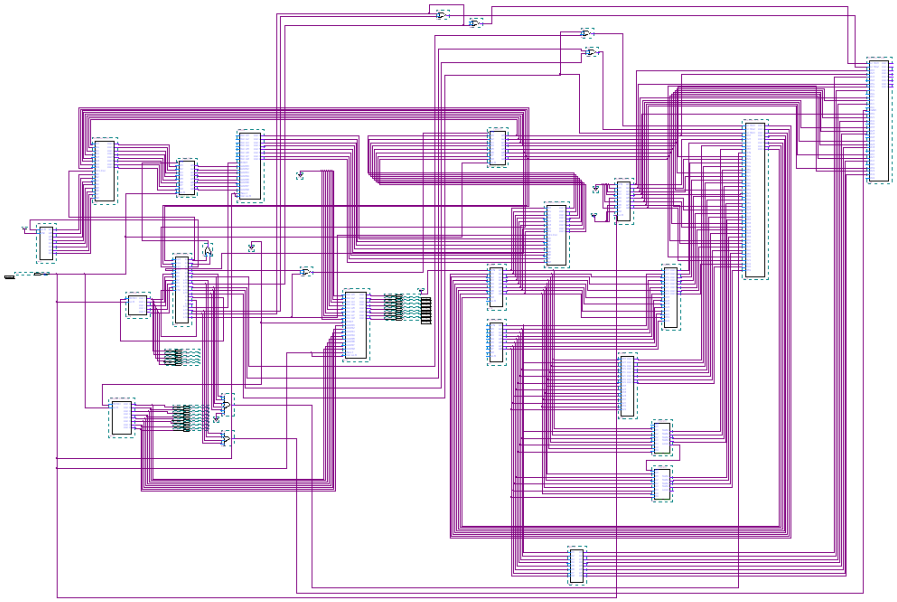
9.1 根据实验逻辑图完成模块设计并测试

我们将简易计算机的设计实现分为 T 计数器设计模块，A,B,R 寄存器设计模块，logicProcess 逻辑设计模块 (包括 IR 和 logicPr2 两个部分的内容)，PC 模块，RAM 模块，选择逻辑模块。该部分的具体内容已经在逻辑设计部分完成，详情参加逻辑设计部分。

9.2 逻辑模块的集成

将上述的多个模块集成在同一文件夹中，并且进行测试通过。

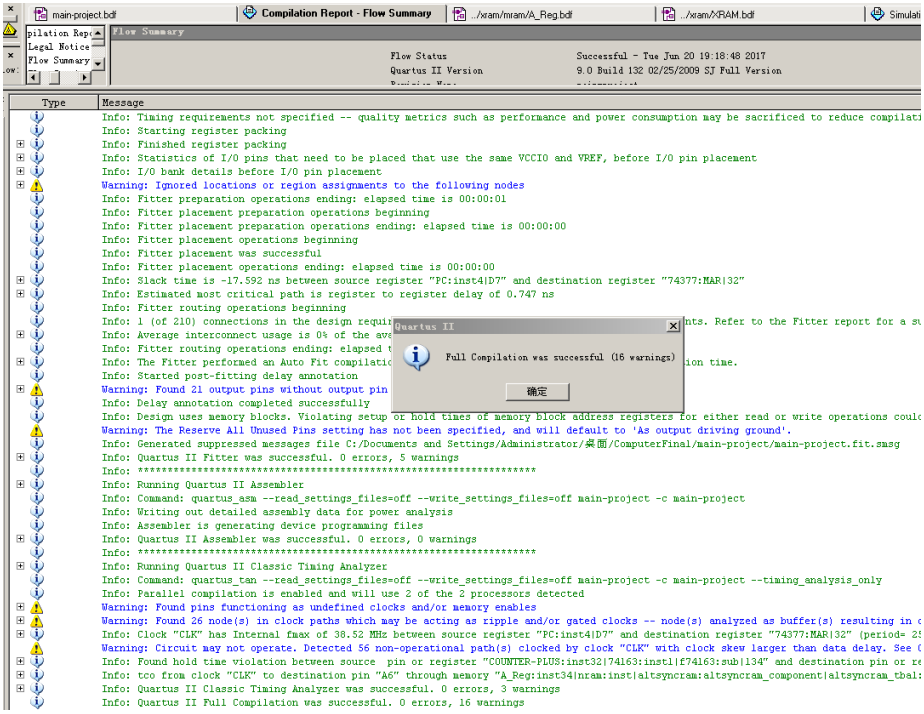
其中，逻辑模块集成之后的逻辑结构图如下所示；



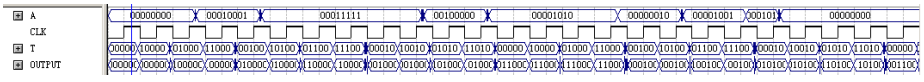
9.3 计算机总逻辑图的编译与调试

经过反复测试与 bug 修正，结果表明，我们的计算机顺利地通过了编译与仿真。

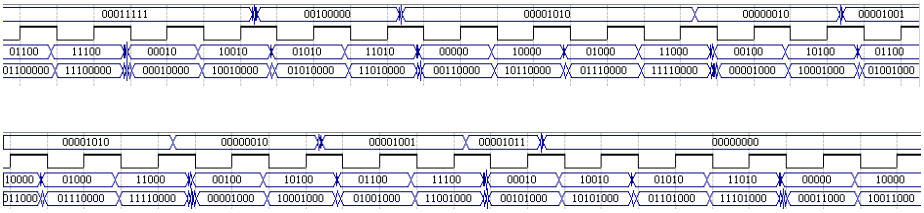
编译完成图片如图所示；



仿真完成图片如图所示：



以下是局部放大波形图：



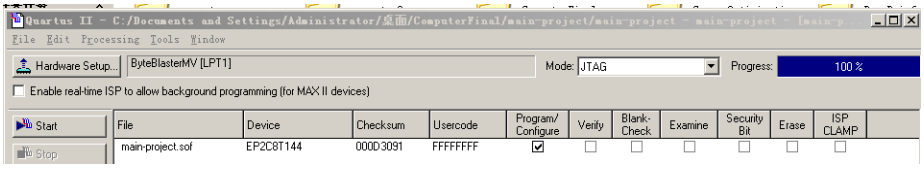
9.4 简易计算机的下载

计算机下载所使用的各输入输出端口以及其对应的引脚如图所示；

Named: * Edit:

	Node Name	Group	Direction	Location
1	A1		Output	PIN_43
2	A2		Output	PIN_44
3	A3		Output	PIN_41
4	A4		Output	PIN_42
5	A5		Output	PIN_32
6	A6		Output	PIN_40
7	A7		Output	PIN_30
8	A8		Output	PIN_31
9	CLK		Input	PIN_75
10	COUNT1		Output	
11	COUNT2		Output	
12	COUNT3		Output	
13	COUNT4		Output	
14	COUNT5		Output	
15	OUT1		Output	
16	OUT2		Output	
17	OUT3		Output	
18	OUT4		Output	
19	OUT5		Output	
20	OUT6		Output	
21	OUT7		Output	
22	OUT8		Output	
23	CLK1		Unknown	PIN_18
24	<<new node>>			

下载完成图片如图所示：



10 实验总结

小半个学期的《数字逻辑实验》课程的学习就这样结束了，同时也应该是张老师教给我们的最后一次课了。当时选定课题的时候，我们小组的五个人毅然决然地选择了计算机，而且有着种种不“简单”的想法，在这个过程去去不断地验证去学习，同时张老师提供了很多方向性的指导，一路走来，受益良多，感慨良多。

我将从**能力提升，意识培养，认识不足，兴趣提升，注重想法，敢于开拓**这六个角度进行归纳总结：

10.1 能力提升

虽然我们是软件工程专业的本科生，但是数字逻辑的学习过程中我们掌握了很多硬件设计的基本方法，同时在对于计算机设计实现的研究过程中，查阅了大量的相关资料，在这一路走来，能力，至少硬件设计以及搜索资料、获取资料的能力得到了较大提升。

10.2 意识培养

张老师以及毛老师上课反复强调，硬件设计是一种思维方法，是一种意识，是大家的按照硬件设计的基本流程进行分析问题，解决问题的意识，这种意识才是比知识更为重要的东西。老师在课堂之上循循善诱，逐步地引导我们对计算机、对硬件设计进行深入理解，同时又是如何去使用最简单的元器件进行硬件设计。在这个过程中，培养了我们的硬件分析与设计意识。

10.3 认识不足

在数字逻辑专题实验课程的学习过程中，这段经历总体而言是非常有趣的，很多的老师精选的例题不仅具有代表性，同时不乏趣味性，让我打开眼界，为之后的简易计算机的设计与实现，以及将来的硬件设计与应用的进一步学习奠定了坚实的基础。

在完成实验报告的过程中，总体思路经过了多次修正，硬件设计也经历了许多的曲折，而且在实际地单独对于某些问题进行系统分析的过程之中，发现实际上遇到了一些阻碍，同时也说明了自己的能力有待提升。虽然只是

简易计算机的设计与实现，但是相对而言实际的计算机设计与实现，更具有挑战性，之后要加强这方面能力的提升以及强化训练。

10.4 兴趣提升

数字逻辑个人认为更多的是在兴趣驱动之下进行分析问题解决问题的过程，在这个过程中，遇到了很多新奇有趣的小技巧，同时也运用了许多让人眼前一亮的方法，极大地提升了自己进一步学习硬件设计，并提升数字逻辑设计能力的兴趣。

10.5 注重想法

数字逻辑的核心在于进行模块化的设计，同时在知晓全局的基础之上进行模块的实现，然后再将各个模块进行集成。往往好的模块规范决定了设计的成败与否，因此需要敢于去提出自己的想法，然后不断地修正，以期臻于完美。

10.6 敢于开拓

简易计算机，或者说硬件设计的方法不是唯一的，或许能够得到满意的结果的都是合理化的方法，但是在这个过程中，如何加强工程化思想的渗透，同时设计得成一门艺术，则需要不断地去修正，不断地去开拓创新，这样才能够不断地完善简易计算机。

同时，其他课程以及科研训练的真谛也在于此，真正的魅力不在于我马上就可以找到解决问题的最好方法，而是在不断地摸索中，提升了个人的能力，培养了开拓创新的精神，这才是社会赖以生存发展的本质意义。

11 参考文献

[1] 鲍家元，毛文林，张琴. 《数字逻辑》(第三版). 高等教育出版社. 2011 年 11 月.

[2] 毛文林. 《数字逻辑课件》(第八章). 数字系统设计. 2017 年 4 月.

[3] 张琴. 《数字逻辑实验课件》. 西安交通大学. 2017 年 4 月.

[4] Intel® 64 and IA-32 Architectures Software Developer’s Manual. Volume 2 (2A, 2B, 2C and 2D): Instruction Set Reference, A-Z

[5] 杜小智. 《嵌入式软件设计》. ARM 指令集. 西安：西安交通大学. 2017 年 3 月.

[6] D.G.Meyer. Design of a Simple Computer. Introduction to Digital System Design. Purdue University.

[7] Atera. https:www.atera.comfaq

[8] A Simple Computer. http:cs.lmu.edu raynotessimplecomputer

[9] Computer Design and Integration: CDI. http:www.cdillc.com

12 附录

12.1 I: 简易计算机指令集

操作码	助记符	执行功能
0 0 0 0 0 0 0 0	LDA addr	将地址 addr 中的内容存入 A 寄存器处
0 0 0 0 0 0 0 1	STA addr	将 A 寄存器中的内容存入地址 addr
0 0 0 0 0 0 1 0	ADD addr	将地址 addr 的内容加上 A 寄存器内容，存入 A 寄存器
0 0 0 0 0 0 1 1	SUB addr	将 A 中内容减去 addr 内容，并将结果存入 A 寄存器
0 0 0 0 0 1 0 0	AND addr	将 addr 中内容与 A 寄存器中的内容进行与操作
0 0 0 0 0 1 0 1	OR addr	将 addr 中的内容与 A 寄存器中的内容进行或操作
0 0 0 0 0 1 1 0	MOV A, R	将 R 中的内容存入 A 寄存器
0 0 0 0 0 1 1 1	MOC B, R	将 R 中的内容存入 B 寄存器
0 0 0 0 1 0 0 0	LDI A, OPRD	将操作数存入 A 寄存器
0 0 0 0 1 0 0 1	LDI B, OPRD	将操作数存入 B 寄存器

表 3: 简易计算机指令集

12.2 II: 内存地址及其对应指令 (基础测试集 1)

R 上电输入 00010101

地址	内容	说明
0000 0000	0000 0110	将 R 中的内容存入 A 寄存器
0000 0001	0000 1001	将下一地址操作数存入 B 寄存器
0000 0010	0000 1110	操作数为 0000 1110
0000 0011	0000 0010	将寄存器 A 与 B 的内容相加，结果存入 A 寄存器
0000 0100	0000 0111	将 R 中的内容存入 B 寄存器
0000 0101	0000 1000	将下一地址操作数存入 A 寄存器
0000 0110	0011 1111	操作数为 0011 1111
0000 0111	0000 0011	将 A 寄存器内容减去 B 寄存器内容存入 A 寄存器
0000 1000	0000 1000	将操作数存入 A 寄存器
0000 1001	0000 1010	操作数 0000 1010
0000 1010	0000 1001	将操作数存入 B 寄存器
0000 1011	0000 0011	操作数 0000 0011
0000 1100	0000 0100	将寄存器 A 与 B 内容相与存入 A 寄存器
0000 1101	0000 1001	将操作数存入 B 寄存器
0000 1110	0000 0101	操作数 0000 1001
0000 1111	0000 0101	将寄存器 A 与 B 内容相或存入 A 寄存器

表 4: 内存地址及其对应指令 1

12.3 III: 内存地址及其对应指令 (基础测试集 2)

R 上电输入 00010101

地址	内容	说明
0000 0000	0000 0110	将 R 中内容 (00010101) 存入 A 寄存器
0000 0001	0000 1001	将下一个地址的操作数存入 B 寄存器
0000 0010	0001 1101	操作数 00011101
0000 0011	0000 0010	将寄存器 A,B 内容相加存入 A 寄存器
0000 0100	0000 0000	进行 LDA addr 操作
0000 0101	1000 0001	在地址 1000 0001 取操作数
0000 0110	0000 0011	将 A,B 寄存器相减的内容存入 A 寄存器
0000 0111	0000 1001	将下一个地址的操作数存入 B 寄存器
0000 1000	0011 1100	操作数
0000 1001	0000 0100	将 A, B 寄存器中内容相与存入 A 寄存器
0000 1010	0000 1000	将下一个地址操作数存入 A 寄存器
0000 1011	0000 0101	操作数
0000 1100	0000 0101	将 A，B 寄存器内容或运算，结果存入 A 寄存器
0000 1101	0000 1000	将下一个地址操作数存入 A 寄存器
0000 1110	1111 1111	操作数
0000 1111	0000 1001	将下一个地址操作数存入 B 寄存器
0001 0000	0101 0101	操作数
0001 0001	0000 0011	将 A，B 寄存器内容相减，结果存入 A 寄存器
—	—	—
1000 0001	0000 0111	操作数 0000 0111
—	—	—

表 5: 内存地址及其对应指令 2

12.4 IV: 微操作表

时间序列	助记符号	注释
标准取指令周期		
t_0	$MAR \Leftarrow PC$	传送指令地址
t_1	Empty	
t_2	$MBR \Leftarrow M[MAR], /WREN, PC \Leftarrow PC + 1$	读出操作码，且 PC+1
t_3	Empty	
t_4	Empty	
t_5	$IR \Leftarrow MBR$	读出操作码，将其传给 IR
t_6	Empty	
—	—	—
MOV A, R	将 R 中的内容存入 A	使用 q1 进行控制
q_1t_7	$A \Leftarrow R, T \Leftarrow 0$	R 内容存入 A，同时 T 清零
q_1t_8	Empty	
—	—	—
MOV B, R	将 R 中的内容存入 B 寄存器	使用 q2 进行控制
q_2t_7	$B \Leftarrow R, T \Leftarrow 0$	R 内容存入 B，同时 T 清零
q_2t_8	Empty	
—	—	—

表 6: 微操作表 1

LDI A, OPRD	将下一地址操作数存入 A	使用 q3 进行控制
q_3t_7	$MAR \Leftarrow PC$	操作数紧跟下一单元
q_3t_8	Empty	
q_3t_9	$MBR \Leftarrow M[MAR], /WREN, PC \Leftarrow PC + 1$	读操作数，PC+1
q_3t_{10}	Empty	
q_3t_{11}	Empty	
q_3t_12	$A \Leftarrow MBR, T \Leftarrow 0$	传送操作数，同时 T 清零
q_3t_{13}	Empty	
—	—	—
LDI B, OPRD	将下一个地址操作数存入 B	使用 q4 进行控制
q_4t_7	$MAR \Leftarrow PC$	操作数紧跟下一单元
q_4t_8	Empty	
q_4t_9	$MBR \Leftarrow M[MAR], /WREN, PC \Leftarrow PC + 1$	读操作数，PC+1
q_4t_{10}	Empty	
q_4t_{11}	Empty	
q_4t_12	$B \Leftarrow MBR, T \Leftarrow 0$	传送操作数，同时 T 清零
q_4t_{13}	Empty	
—	—	—

表 7: 微操作表 2

AND A, B	将寄存器 A 和寄存器 B 内容相与存入寄存器 A	使用 q5 进行控制
q_5t_7	$A \Leftarrow AandB, T \Leftarrow 0$	
q_5t_8	Empty	
q_5t_9	Empty	
q_5t_{10}	Empty	
—	—	—
OR A, B	将寄存器 A 和寄存器 B 内容相或存入寄存器 A	使用 q6 进行控制
q_6t_7	A $\Leftarrow AorB, T \Leftarrow 0$	
q_6t_8	Empty	
q_6t_9	Empty	
q_6t_{10}	Empty	
—	—	—
ADD A, B	将寄存器 A 和寄存器 B 内容相加存入寄存器 A	使用 q7 进行控制
q_7t_7	$A \Leftarrow A + B, T \Leftarrow 0$	
q_7t_8	Empty	
q_7t_9	Empty	
q_7t_{10}	Empty	
—	—	—
SUB A, B	将寄存器 A 和寄存器 B 内容相减存入寄存器 A	使用 q8 进行控制
q_8t_7	$B \Leftarrow -B$	将 B 取反
q_8t_8	$Empty$	
q_8t_9	$B \Leftarrow B + 1$	将 B 取补
q_8t_{10}	$Empty$	
q_8t_{11}	$A \Leftarrow A + B, T \Leftarrow 0$	
q_8t_{12}	$Empty$	
q_8t_{13}	Empty	
q_8t_{14}	Empty	
—	—	—

表 8: 微操作表 3

LDA ADDR	将地址 addr 中的内容存入 A	使用 q9 进行控制
q_9t_7	$MAR \Leftarrow PC$	将下一地址存入 A 中
q_9t_8	Empty	
q_9t_9	$MBR \Leftarrow M[MAR], PC = PC + 1$	读操作数，PC+1
q_9t_{10}	Empty	
q_9t_{11}	Empty	
q_9t_{12}	$MAR \Leftarrow MBR$	传递操作数地址给 MAR
q_9t_{13}	Empty	
q_9t_{14}	$MBR \Leftarrow M[MAR]$	读取操作数
q_9t_{15}	Empty	
q_9t_{16}	Empty	
q_9t_{17}	$A \Leftarrow MBR, T \Leftarrow 0$	传递操作数，转入 T0 取址
q_9t_{18}	Empty	
—	—	—
STA ADDR	将 A 内容存入地址 addr	使用 q10 进行控制
$q_{10}t_7$	$MBR \Leftarrow A$	将 A 寄存器内容存入 MBR
$q_{10}t_8$	Empty	
$q_{10}t_9$	$MAR \Leftarrow PC$	将 PC 地址传给 MAR
$q_{10}t_{10}$	Empty	
$q_{10}t_{11}$	$M[MAR] \Leftarrow MBR$	将 MBR 中内容写入 MAR 地址处
$q_{10}t_{12}$	Empty	
$q_{10}t_{13}$	Empty	

表 9: 微操作表 4

12.5 V: 集成化微操作序列

Xi	微操作序列
$x_1 = t_0 + q_3t_7 + q_4t_7 + q_9t_7 + q_{10}t_9$	$MAR \Leftarrow PC$
$x_2 = q_9t_{12}$	$MAR \Leftarrow MBR$
$x_3 = t_2 + q_3t_9 + q_4t_9 + q_9t_9$	$PC \Leftarrow PC + 1$
$x_4 = t_2 + q_3t_9 + q_4t_9 + q_9t_9 + q_9t_{14}$	$MBR \Leftarrow M[MAR]$
$x_5 = q_3t_{12} + q_9t_{17}$	$A \Leftarrow MBR$
$x_6 = q_1t_7$	$A \Leftarrow R$
$x_7 = q_1t_7 + q_2t_7 + q_3t_{12} + q_4t_{12} + q_5t_7 + q_6t_7 + q_7t_7 + q_8t_{11} + q_9t_{17}$	$T \Leftarrow 0$
$x_8 = t_5$	$IR \Leftarrow MBR$
$x_9 = q_{10}t_{11}$	$M[MAR] \Leftarrow MBR$
$x_{10} = t_2 + q_3t_9 + q_4t_9$	$WREN$
$x_{11} = q_2t_7$	$B \Leftarrow R$
$x_{12} = q_4t_{12}$	$B \Leftarrow MBR$
$x_{13} = q_5t_7$	$A \Leftarrow AandB$
$x_{14} = q_6t_7$	$A \Leftarrow AorB$
$x_{15} = q_8t_7$	$B \Leftarrow B$
$x_{16} = q_8t_9$	$B \Leftarrow B + 1$
$x_{17} = q_8t_{11}$	$A \Leftarrow A + B$
$x_{18} = q_{10}t_7$	$MBR \Leftarrow A$

表 10: 集成化微操作序列