计算机体系设计实践报告

2021 学年秋季学期

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **序号** | **学号** | **姓名** | **专业** | **班级** | **成绩** |
| 245 | 20211060245 | 陈俊宏 | 人工智能 | 人工智能 |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

指导教师：武丽雯

软件学院 计算机体系设计实践 王逍

二 O 二二年十一月

# 计算机体系设计实践

**实验目的**

实践课设计 4 个实验，基本涵盖了计算机体系结构与组成课程的重点内容，实验电路的 设计与教材保持一致，实验电路在 FPGA 实验板上实现，并使用调试软件 JULAB 完成实验 的调试与分析。

**实验要求**

实验课前复习相关原理，按实验指导书的要求认真预习。 实验时独立思考，掌握实验设备或软件的构造和操作方法，按实验指导书要求设计或验

证实验内容，测试有关数据，分析相应的问题。

实验课结束时需要整理提交 fpga 工具箱，并检查工具箱附件是否缺失。

由于 fpga 工具箱数量有限，实验以小组形式完成，2 人 1 个小组，提交 1 份实验报告。

**实验资料环境和资料**

Win10+FPGA 设计软件 Quartus II、实验调试软件 Julab、实验相关设计文件（工程模板、 电路设计源文件、虚拟构图文件）等。请找实验知道老师索取。

**实验地点**

计算机硬件实验室：软件学院大楼 1428 室

# 实验 1 加减运算及特征标志

## 实验操作

1. 下载实验资源

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。将加减 运算及特征标志中文件解压到 DE2-115 工程文件夹中；

解压后的 *lab2.vpl* 和 *lab2.bmp* 是留给实验调试软件使用的虚拟面板构图文件。

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。

点击工具栏中分析与综合（Start Analysis & Synthesis）按钮，检查语法错误，参阅实验 指导书第五章 5.1.1 设计流程的“分析综合”。

分析综合通过后，直接点击工具栏中的全编译（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 *lab2.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，填写实验结果记录和分析。

## 实验记录

1. 运算功能和控制信号

根据实验原理分析各种运算对应的控制信号，填入下表。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 运算指令 | 运算功能 | 运算控制信号 | | | |
| M3 | M2 | M1 | M0 |
| ADD | F=dst+src | 0 | 0 | 0 | 1 |
| SUB | F=dst-src | 0 | 1 | 1 | 1 |
| ADDC | F=dst+src+进位 | 1 | 0 | 0 | 1 |
| SUBB | F=dst-src-借位 | 1 | 0 | 1 | 0 |
| INC | F=dst+1 | 0 | 1 | 0 | 0 |
| DEC | F=dst-1 | 0 | 0 | 1 | 0 |
| 无 | F=dst | 0 | 0 | 0 | 0 |

2. 数据传送

设置 M3～M0 实现数据传送，使加法器的输出 F=A。下表中双线左侧是输入信号，右 侧是输出信号。按照表中给出的输入数据，通过拨动开关送给 FPGA 实验电路；将相关指示 灯的结果，填入表格右部栏目。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dst | src | Ci | M3~M0 | B | C0 | F | 实验现象分析 |
| ① | 1010 | 1111 | — | 0000 | 0 | 0 | A | 如果改变 src 的值，对 B 和 C0 的 值 没有 （有/没有）影响。 |
| ② | 1010 | 0000 | — | 0000 | 0 | 0 | A |

要将 dst 输入端的数据送到加法器的 F 输出端，需要使 M3~M0= 0 ，这时 B= 0 、 C0= 0 ，因此 F = A。

3. 加法运算结果的特征标志

设置 M3~M0 为加法运算，按下表步骤操作，观察加法运算的结果，填入下表，并写出 计算数和结果的真值。

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dst | src | Ci | M3~M0 | F | FLAG | | | | 运算数和运算结果的真值 | |
| S | Z | O | C | 视为无符号数 | 视为补码 |
| ① | 1000 | 0001 | — | 0001 | 1001 | 1 | 0 | 0 | 0 | 8＋1＝9 | (**﹣**8)＋1＝**﹣**7 |
| ② | 1101 | 1100 | — | 1001 | 1 | 0 | 0 | 1 | 13+12=25 | (-3)+(-4)=(-7) |
| ③ | 0100 | 0010 | — | 0110 | 0 | 0 | 0 | 0 | 4+2=6 | 4+2=6 |
| ④ | 0000 | 0000 | — | 0000 | 0 | 1 | 0 | 0 | 0+0=0 | 0+0=0 |
| ⑤ | 1111 | 0001 | — | 0000 | 0 | 1 | 0 | 1 | 15+1=16 | (-1)+1=0 |
| ⑥ | 0011 | 0101 | — | 1000 | 1 | 0 | 1 | 0 | 3+5=8 | 3+5=8 |
| ⑦ | 1100 | 1011 | — | 0111 | 0 | 0 | 1 | 1 | 12+11=23 | (-4)+(-5)=(-9) |
| ⑧ | 1100 | 0101 | — | 0001 | 0 | 0 | 0 | 1 | 12+5=17 | (-4)+5=1 |
| ⑨ | 0011 | 1011 | — | 1110 | 1 | 0 | 0 | 0 | 3+11=14 | 3+(-5)=(-2) |
| ⑩ | 1000 | 1000 | — | 0000 | 0 | 1 | 1 | 1 | 8+8=1 | (-8)+(-8)=(-16) |

提示：为方便分析运算结果，可以事先列出负数的 4 位补码与真值的对应关系：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| -8 | -7 | -6 | -5 | -4 | -3 | -2 | -1 |

实验现象分析：

（1）负标志 SF 就是运算结果的　最高位 （最高位／最低位）。

（2）零标志 ZF 的生成和　Ｆ （F／CF／F 及 CF）有关。

（3）溢出标志 OF 和进位标志　没有 （有／没有）直接的联系。

（4）对照标志位和真值，可以看出溢出标志 OF 是按照　补码 （无符号数／补码） 的运算结果设置的；进位标志 CF 是按照　无符号数　（无符号数／补码）运算的结果设置的。

也就是说，如果运算数是无符号数，运算结果是否溢出是由 CF　（CF／OF）反映的；如果 运算数是有符号补码数，运算结果是否溢出是由　OF　（CF／OF）反映的。

（5）4 位补码能表示数值的范围是　(－８～７)　，4 位无符号数能表示数值的范围是

　 (０～15)　。

（6）运算器电路是否“知道”运算数是有符号数还是无符号数？　运算器电路无法区分

4. 减法运算

减法运算是转换为加法计算的。设置 M3~M0 为减法运算，注意观察 B 操作数、C0 和

FLAG 的 CF（进位）标志位。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dst | src | Ci | M3~M0 | C0 | B | F | CF | 实验现象分析 |
| ① | 0010 | 0001 | — | 0111 | 1 | 1110 | 0001 | 1 | 无　(有/无)借位 |
| ② | 0001 | 0010 | — | 1 | 1101 | 1111 | 0 | 有　(有/无)借位 |

实验现象分析：

———

（1） 减法运算时，B= 　　（src/src），C0= １　（1/Ci），所以 F=　+2

（2） CF 标志与减法运算有没有产生借位　有　(有/没有)关系，没有产生借位时，

CF=　1　；减法运算产生借位时，CF=　0　。

5. 带借位的减法运算

设置 M3~M0 为带借位的减法运算，注意观察 F 和 Ci 的关系。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dst | src | Ci | M3~M0 | C0 | B | F | CF | 实验现象分析 |
| ① | 0101 | 0011 | 1 | 1010 | 1 | 1111 | 0101 | 1 |  |
| ② | 0101 | 0011 | 0 | 0 | 1111 | 0100 | 1 |  |

实验现象分析：

在带借位的减法运算中，Ci 代表的是　借位的逻辑反　（借位/借位的逻辑反）。从实验结果可以看 出，当 Ci=1 时，F=dst – src –　0　(1/0)；当 Ci=0 时，F=dst – src – 　1　 (1/0)。请解释这个实验结果：　由实验得到的表格可以看出，这个电路就是用来实现1位二进制的被减数、减数和低位来的接位数三者相减的电路。

6. 加 1 和减 1 运算

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | dst | src | Ci | M3~M0 | C0 | B | F | FLAG |
| ① INC | 0010 | 0101 | 1 | 0100 | 0 | 0000 | 0011 | 0000 |
| ① INC | 0010 | 1010 | 0 | 0 | 0000 | 0011 | 0000 |
| ②  DEC | 0010 | 1010 | 0 | 0010 | 0 | 1111 | 0001 | 0001 |
| ②  DEC | 0010 | 0101 | 1 | 0 | 1111 | 0001 | 0001 |

实验现象分析：

（1）加 1 运算时，B 始终为　0000　，C0 始终为　0　，所以 F = A+B+C0 =Ａ＋１。

（2）减 1 运算时，B 始终为　1111　即-1，C0 始终为　0　，所以 F = A+B+C0 =Ａ－１。

（3）改变 src 的值，对结果　没有　(有/没有)影响。

## 实验小结及实验分工

实验小结：

答：实验存在一定的难度，尤其是在统计数据的时候十分容易出错，一不小心就会造成“差之毫厘谬以千里的”下场，所以在试验过程中需要十分的细心仔细，认真核对每一个数据和结果，才能最终总结出正确的答案。

实验分工：单人完成

# 实验 2 高速缓冲存储器

实验操作

1. 下载实验资源

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。将高速 缓冲存储器中文件解压到 DE2-115 工程文件夹中；

实验电路顶层文件 *Lab\_Top.v*、地址译码 *Decoder.v、*地址寄存器 *R.v*、多路器 *MUX.v*、

VALID 模块 *ram\_valid.v*。

*lab5.vpl* 和 *lab5.bmp* 是留给实验调试软件使用的虚拟面板构图文件。

init\_mm.mif 是主存 MM 内容的初始化文件。

Q12 文件夹里是使用 FPGA 内部的 RAM 资源设计的主存 MM 模块 *ram\_mm.v*、CACHE

模块 *ram\_cache.v*、标志存储器 TAG 模块 *ram\_tag.v* 以及它们各自的 IP 核文件。

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。

点击工具栏中分析与综合（Start Analysis & Synthesis）按钮，检查语法错误，参阅实验 指导书第五章 5.1.1 设计流程的“分析综合”。

分析综合通过后，直接点击工具栏中的全编译（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 *lab5.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。

本实验验证时需要使用 QuartusII 软件的在系统存储器数据编辑器（In-System Memory Content Editor）， 实时查看和修改标志存储器 TAG、高速缓存存储器 CACHE 和主存 MM 的内容，In-System Memory Content Editor 的更多使用方法，参阅实验指导书第五章 5.1.3 在 系统存储器数据编辑器。

## 实验记录

1. 主存地址格式各部分的位数。

AR

|  |  |  |
| --- | --- | --- |
| TAG | BLOCK | WORD |

2. 初始状态

使用 Quartus II 的 In-System Memory Content Editor 查看 TAG、CACHE 和 MM 的内容， 并对后面用到的主存 50H~53H、64H~67H、84H~87H 单元输入一些已知的内容，记录在下 表中。

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 地址 | 50H | 51H | 52H | 53H | 64H | 65H | 66H | 67H | 84H | 85H | 86H | 87H |
| 内容 | 0 | 1 | 2 | 3 | 0 | 5 | 6 | 7 | 0 | A | B | C |

观察 8 个 VALID 单元的状态应都为 0，如果不是，按 RESET 键清零。（实验原理图上

没有画出 RESET 按键与 VALID 模块的连接）

3. 不命中情况下 CACHE 内容的装入

因为是直接映像，映射关系已经固定，主存中的某一块只能存入 cache 的指定位置，所 以不需要考虑替换算法，不命中时直接装入即可。装入时需要依次装入 4 个字，由 OFFSET 选择写入哪一个字，WR\_CACHE 给出读 MM 和写 CACHE 的时钟。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AB | CLK | OFFSET | WR\_CACHE | MM\_DATA | WR0 | WR1 | WR2 | WR3 | HIT |
| ① | 52H |  | **——** | 0 | **——** | 0 | 0 | 0 | 0 | 0 |
| ② | 52H | 0 | 00 |  | 1 | 1 | 0 | 0 | 0 | 0 |
| ③ | 52H | 0 | 01 |  | 2 | 0 | 1 | 0 | 0 | 0 |
| ④ | 52H | 0 | 10 |  | 3 | 0 | 0 | 1 | 0 | 0 |
| ⑤ | 52H | 0 | 11 |  | 4 | 0 | 0 | 0 | 1 | 1 |

上述操作完成后，用 In-System Memory Content Editor 查看 TAG 和 CACHE 中变化的内 容记录在下表中。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 行号 | TAG | DATA0 | DATA1 | DATA2 | DATA3 |
| 4 | 00002000 | 00001000 | 00002000 | 00003000 | 00004000 |

该行的 V =　1　（0/1）。

实验分析：

（1）52H 的地址访问的是 CACHE 的第　4　行第　0　个字。

（2）当所访问的地址不命中时，需将访问地址所指向的主存块的　所有单元　（一 个单元／所有单元）装入 CACHE。

（3）在向 CACHE 存储器中写入第　３　（0／1／2／3）个字的时候，TAG 存储器、

VALID 存储器也同时写入。

4. 命中情况下 CACHE 的读出

访问 50H，51H，52H，53H 地址，这 4 个地址对应着同一个主存块中的 4 个单元，在 上一步操作中，访问 52H 地址不命中后，访问地址所指向的主存块已经整个装入了 CACHE 块，所以访问该主存块中的任意单元，应该都是命中的，直接从 CACHE 读出。

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AB | CLK | WR\_CACHE | AR- 区号 | TAG | HIT | AR - 字地址 | CACHE\_WORD |
| ① | 50H |  | 0 | 010 | 010 | 1 | 00 | 1 |
| ② | 51H |  | 0 | 010 | 010 | 1 | 01 | 2 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
| ③ | 52H |  | 0 | 010 | 010 | 1 | 10 | 3 |
| ④ | 53H |  | 0 | 010 | 010 | 1 | 11 | 4 |

访问某一主存单元时，根据地址寄存器 AR 的　块号　（区号／块号／字地址）找到

CACHE、TAG 和 VALID 的行；如果该行的 TAG 与地址寄存器 AR 的　区号　（区号／块号

／字地址）相同，并且 VALID=　１　（0/1），则要访问的主存地址命中，判断是否命中的代 码在**错误!未找到引用源。**的第　？　行；命中时根据地址寄存器 AR 的　字地址　（区号／块号／字地址）由多路器 MUX 选择读出 CACHE 行中的哪一个字，多路器 MUX 的实例化在**错 误!未找到引用源。**的第　？　行。

5. 抖动现象

直接映像方式下，每个主存块都只有固定的一个 cache 位置可以存放，当主存地址的区 内块号相同的时候，由于对应同一个 cache 块，即便其他 cache 块都是空闲，也无法使用。

当某段时间内恰巧要访问主存不同区号但相同区内块号的两块数据时，例如下面第一个 表格中 1~8 行的地址 84H～87H，与 9~16 行的地址 64H～67H，分别属于主存的第 101　区和第　100　区，区号不同，但它们的区内块号相同，都是　001　，如果 CPU 交替访问这 两块数据，就会出现这两块主存数据交替调入调出 CACHE 的现象，这种现象称为抖动。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | AB | CLK | OFFSET | WR\_CACHE | MM\_DATA | AR- 区号 | AR - 块号 | AR - 字地  址 | HIT | CACHE\_WORD |
| 1 | 84H |  | **——** | 0 | **——** | 101 | 001 | 00 | 0 | **——** |
| 2 | 84H | 0 | 00 |  | 0 | （同上） | （同上） | （同上） | 0 | **0** |
| 3 | 84H | 0 | 01 |  | 0 | （同上） | （同上） | （同上） | 0 | **——** |
| 4 | 84H | 0 | 10 |  | 0 | （同上） | （同上） | （同上） | 0 | **——** |
| 5 | 84H | 0 | 11 |  | 0 | （同上） | （同上） | （同上） | 1 | **——** |
| 6 | 85H |  | **——** | 0 | **——** | 101 | 001 | 01 | 1 | 0 |
| 7 | 86H |  | **——** | 0 | **——** | 101 | 001 | 10 | 1 | 0 |
| 8 | 87H |  | **——** | 0 | **——** | 101 | 001 | 11 | 1 | 0 |
| 9 | 64H |  | **——** | 0 | **——** | 100 | 001 | 00 | 0 | **——** |
| 10 | 64H | 0 | 00 |  | 0 | （同上） | （同上） | （同上） | 0 | A |
| 11 | 64H | 0 | 01 |  | 0 | （同上） | （同上） | （同上） | 0 | **——** |
| 12 | 64H | 0 | 10 |  | 0 | （同上） | （同上） | （同上） | 0 | **——** |
| 13 | 64H | 0 | 11 |  | 0 | （同上） | （同上） | （同上） | 1 | **——** |
| 14 | 65H |  | **——** | 0 | **——** | 100 | 001 | 01 | 1 | B |
| 15 | 66H |  | **——** | 0 | **——** | 100 | 001 | 10 | 1 | C |
| 16 | 67H |  | **——** | 0 | **——** | 100 | 001 | 11 | 1 | D |
| 17 | 84H |  | **——** | 0 | **——** | 101 | 001 | 00 | 0 | **——** |

## 实验小结及实验分工

答:由于实验过程中对于步骤的不清晰，花了很多时间在对仿真软件的操作方式的摸索中，虽然开始未有成果，但是对cache的存储过程有了一定的了解，在有同学解决仿真软件存在的问题后，很快就完成了实验，由此可知：基础是十分重要的，基础牢固后，实践随之轻而易举。

实验分工：单人完成

# 实验 3 指令和寻址方式

实验操作

1. 下载实验资源

解压缩指令和寻址方式，得到两个文件，*Lab\_JUC2.sof* 是用来下载到 DE2-115 的实验模 型机 JUC2 电路文件，*JUC2.scc* 是给实验调试软件用的 CPU 配置文件。

2. 实验电路下载

点击桌面上的 QuartusII Programmer 图标，打开 QuartusII 编程器，点击添加编程文件按 钮（Add File），浏览选择前面下载的实验模型机 JUC2 电路文件 *Lab\_JUC2.sof*，点击 Start 按钮，下载到 DE2-115 实验板。

3. 验证实验电路功能

打开实验调试软件 JULAB3，选择模型计算机实验类型，在“CPU 数据通路”窗口，可 以看到模型机 JUC2 的数据通路图。在“文件”菜单下，选择“打开 CPU 配置”选项，浏 览选择前面下载的 *JUC2.scc*，文件中载入的观察信号出现在调试软件的“寄存器及总线信息 窗口”。

实验的验证需要使用“主存汇编/调试窗口”为模型机主存写入机器指令，在该窗口输 入的汇编指令会自动翻译成对应的机器指令并写入主存；完成的汇编指令可以导出到文件中 保存，也可以直接导入已有的汇编文件；更多详细内容参阅实验指导书第 5 章 5.4.2 实验系 统软件中的“主存汇编/调试窗口”内容。

实验的验证需要使用“主存信息显示窗口”以十六进制形式手工输入数据到模型机主存； 点击工具栏中的主存刷新按钮，可以在该窗口查看模型机主存各单元内容；主存内容可以导 入或导出；更多详细内容参阅实验指导书第 5 章 5.4.2 实验系统软件中的“主存信息显示窗 口”内容。

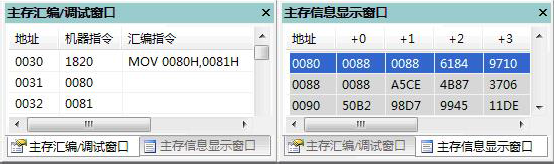
实验的验证要以机器指令单步的方式执行写入到模型机主存中的指令，点击工具栏中的 指令单步按钮执行指令，如果出错或想重新从头开始调试，工具栏中的复位按钮可以对模型 机硬件电路进行复位，更多调试方法参阅实验指导书第 5 章 5.4.2 实验系统软件中的“运行 及调试”内容。

## 实验记录

1. 基本寻址方式

将下面表格中的指令通过“主存汇编/调试窗口”输入到模型机的主存，以表格中的第 一条指令 MOV 0080H,0081H 为例，如下图左；执行前，在“主存信息显示窗口”中，为 主存的 0080H 单元手工输入数据 0088H，输入完成后以“指令单步”方式运行，“主存汇

编/调试窗口”会动态跟踪执行的机器指令，可以观察到光标移动到了下一条即将执行的指 令处。刷新主存内容，如果指令执行成功，在“主存信息显示窗口”的 0081H 单元，可以 看到指令的执行将 0080H 单元的数据 0088H 成功传送到了 0081H 单元，如下图右。



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 指令 | 执行前数据 | 执行后数据 | 结果分析 |
| ① | MOV 0080H, 0081H | (0080H)= 0088H (0081H)= 0000H | (0081H)= 0088H | 两个操作数的寻址方式都是直接寻址　该指令的功能是将　0080H　单元的内容传送到　0081H　单元 |
| ② | MOV #0080H, R0 | (R0)= 0000H | (R0)= 0080H | 源操作数的寻址方式是　立即寻址　，立即数包含在指令中，所在单元的地址是　8800H　，R0的内容即来自于该单元 |
| ③ | MOV (0080H), R1 | (0080H)= 0088H (0088H)= 0082H | (R1)= 0082H | 源操作数的寻址方式是　间接寻址　，0080H单元存放的是　有效地址　，R1寄存器内容是主存　0088H　单元的内容 |
| ④ | MOV (R1), R2 | (R1)= 0082H  (0082H)= 0000H | (R2)= 0000H | 源操作数的寻址方式是　间接寻址　，R1寄存器的内容是　有效地址　，R2寄存器的内容是主存　0082H　单元的内容 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ⑤ | MOV 8(R0), 0082H | (R0)= 0080H  (0082H)= 0000H | (0082H)= 0082H | 源操作数的寻址方式是　变址寻址　，有效地址的计算方法是　0080H+8H　，0082H单元的内容是主存　0088H　单元的内容 |

2. 移位、条件转移指令和相对寻址

将下面汇编语言程序手工翻译成机器指令，填写在横线上，在“主存信息显示窗口”， 将翻译好的机器指令，输入到模型机的主存，以“指令单步”方式运行。

1

2

3

4

5

6

7

ORG 0030H

0030: 1601 0505 ; MOV #0505，R1

0032: 6601 0001 ; AND #0001，R1

0034: 0378 0001 ; JNZ 1(PC)

0036: 0000 ;

0037: 0101 ;

HALT

ROL R1

0038: 0420 0032 ; JMP 0032H

下表已经给出了开始几条指令运行记录的内容，在后面的空白行上记录后续执行的指令

行号以及执行后的相关数据（如相关寄存器和 PSW 的变化），分析执行结果的意义（如程 序是否转移，转移的目的地址是多少），直到运行到 HALT 指令。

|  |  |  |
| --- | --- | --- |
| 指令 行号 | 指令执行后相关数据 | 结果分析 |
| 02 | (R1) = 0505H |  |
| 03 | (PSW) = 0  (R1) = 0001H | PSW 中的零标志位 Z= 0 。 |
| 04 | (PC) = 0037H | 发生 （发生／不发生）转移。相对寻址的有效地址  EA = (PC) + 偏移量，该指令计算有效地址时(PC) =  0037H，所以转移的目的地址是 0037H 。 |

3. 入栈和出栈指令

将下面汇编语言程序输入到模型机的主存，输入时注意根据指令的字长确定每条指令所 在的主存地址，以“指令单步”方式运行。观察堆栈指针 SP、堆栈存储单元以及相关寄存 器和内存单元的变化，记录在下表中，理解堆栈的用法。

。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 指令 | 执行前数据 | 执行后数据 | 结果分析 |
| ① | MOV #0041H，R0 |  | (R0)= 0041H |  |
| ② | PUSH R0 | (R0) = 0041H  (SP) = 0030H (002F) = 0000H | (SP) = 002F H (002F) = 0041H | 堆栈的第一个数据存放在主存0041H单元，其地址存放在R0  寄存器中。 |
| ③ | PUSH 0040H | (0040H) = 5555H (SP) = 002F H (002E) = 0000H | (SP) = 002E H  (002E) = 5555 H | 堆栈空间是朝着地址 减小／增 大 方向增长的，称作向上增长 |
| ④ | POP (R0) | (SP) = 002E H  (R0)= 0041H (0041H)= 0000H | (SP) = 002F H  (R0)= 0041H (0041H)= 5555H | 堆栈遵循 先进先出／后进先出 的原则。0041H 单元的内容是原 来 002E H 单元的内容。 |
|  |  |  |  |  |
| ⑤ | POP R1 | (SP) = 002F H  (R1)= 0000H | (SP) = 0030H  (R1)= 0000H | R1 的内容是原来 002F H 的内容。 |

4. 子程序调用和返回

下面的程序将 0038H 单元的内容读入寄存器 R1，调用子程序完成乘 2，返回主程序后 将结果保存到 0039H 单元。程序运行前需要先设置 0038H 单元的值，。

|  |  |  |  |
| --- | --- | --- | --- |
| 1 |  |  | ORG 0030H |
| 2 | 0030: | ; | MOV 0038H, R1 |
| 3 | 0031: |  |  |
| 4 | 0032: | ; | CALL 0040H |

|  |  |  |  |
| --- | --- | --- | --- |
| 5 | 0033: |  |  |
| 6 | 0034: | ; | MOV R1, 0039H |
| 7 | 0035: |  |  |
| 8 | 0036: | ; | HALT |
| 9 |  |  | ORG 0040H |
| 10 | 0040: | ; | ADD R1, R1 |
| 11 | 0041: | ; | RET |

将上面的程序输入到模型机，将机器码填入横线上。单步运行，观察子程序调用和返回

前后的堆栈变化，填写下面的表格。

|  |  |  |  |
| --- | --- | --- | --- |
| 指令行号 | 执行前数据 | 执行后数据 | 结果分析 |
| 02 | (R1)= 0000H  (0038H)= 5050H | (R1)= 5050H | R1 的内容和 0038H 单元的数据一致。 |
| 04 | (SP) = 002F H (002F) = 0000H (PC) = 0040H | (SP) = 002F H (002F) = 0034H (PC) = 0041H | 执行后堆栈中存放的是返回地址，即  CALL 指令下面一条指令的地址。 |
| 11 | (SP) = 0030H  (PC) = 0036H | (SP) = 0030H  (PC) = 0036H | 执行后的 PC 内容来自于堆栈的栈顶单元， 即返回到 CALL 指令下面一条指令。 |
| 06 | (R1)= A0A0H | (0039H = A0A0 H | 0039H 单元的数据即 R1 寄存器的内容， 是 0038H 单元数据的 2倍 。 |

## 实验小结及实验分工

实验分工：单人完成

实验小结：实验要求掌握基本寻址方式；移位、条件转移指令和相对寻址；入栈和出栈指令；子程序调用和返回四个方面的应用原理，需要具备一定量的基础理论知识，实验本身来说只要细心且有耐心，不难完成。

# 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 I9~6 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。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RESET | Clock | I9~0 | CP1 | CP2 | µAR | CMdata(H) | BM | NA | 有效的控制信号 |
|  |  | —— | 0000000000 | 0 | 0 | —— | —— | —— | —— | —— |
| ① | 0 |  | —— | 0 | 1 | 00H | 10001 | 0 | 0 | —— |
|  | 0 |  | —— | 1 | 0 | —— | —— | —— | —— | IRce |
| ② | 0 |  | —— | 0 | 1 | 01H | 00042 | 1 | 1 | BM1 |
|  | 0 |  | —— | 1 | 0 | —— | —— | —— | —— | —— |
| ③ | 0 |  | —— | 0 | 1 | 02H | —— | —— | —— | —— |

实验结果分析：

复位时，CP2=　0　，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　CP2　（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　CP2　（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　CP1　（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 uAR=　01H　，表明第①条微指令的微地址转移方式为　计数器方式　，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=　02H　，表明第②条微指令的微程序转移方式为　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　运算指令　（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　00042H　。 将指令寄存器输入端的开关 I9~6 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=　02H　，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　00042H　。 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　08H　。

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（ IR9~IR6）是 0000B，INDEX（IR5~IR4）是 01B，DATA

（IR3~IR0）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 | 02 | 64006 | 11 | 001 | 0000 | 00 | 00 | 000110 | DATAoe　　GRSce |
| 观察 | 06 | 20000 | 10 | 000 | 0000 | 00 | 00 | 000000 | GRSoe |

（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。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | RESET | Clock | CP1 | CP2 | µAR | CMdata | 有效的控制信号 | BUS | INDEX |
|  |  | —— | 0 | 1 | —— | —— | —— | —— | —— |
| ① | 0 |  | 1 | 0 | 00H | 10001H | —— | —— | —— |
|  | 0 |  | 0 | 1 | —— | —— | IRce | —— | —— |
| ② | 0 |  | 1 | 0 | 01H | 00041H | IRce | —— | 01 |
|  | 0 |  | 0 | 1 | —— | —— | —— | —— | 01 |
| ③ | 0 |  | 1 | 0 | 02H | 64006H | —— | —— | 01 |
|  | 0 |  | 0 | 1 | —— | —— | DATAoe　　GRSce | 5 | 01 |
| ④ | 0 |  | 1 | 0 | 06H | 20000H | 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 R 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 行修改为

。 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（ IR9~IR6）是　0001H　，INDEX（IR5~IR4）是　01　， DATA（IR3~IR0）是　0111　，因此翻译出指令机器码是　0001010111　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） 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　第35、36行　，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004H~007H　。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 03 | 28004 | 01 | 010 | 0000 | 00 | 00 | 000100 | GRSoe　　Ace |
| 04 | 00080 | 00 | 000 | 0000 | 00 | 10 | 000000 | 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 | 11 | 6c705 | 11 | 100 | 0001 | 11 | 00 | 000101 | DATAoe　PSWce　ADD |
| 保存 | 05 | 44006 | 10 | 001 | 0000 | 00 | 00 | 000110 | Soe　　GRSce |
| 观察 | 06 | 20000 | 01 | 000 | 0000 | 00 | 00 | 000000 | GRSoe |

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 | µAR | CMdata | 有效的控制信号 | A | BUS | S\_Q | PSW | INDEX |
|  |  | —— | —— | —— | —— | —— | —— | —— | —— | —— |
| ① | 0 |  | 00 | 10001 | —— | —— | —— | —— | —— | —— |
|  | 0 |  | —— | —— | IRce | —— | —— | —— | —— | —— |
| ② | 0 |  | 01 | 00042 | IRce | —— | —— | —— | —— | 01 |
|  | 0 |  | —— | —— | —— | —— | —— | —— | —— | 01 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ③ | 0 |  | 03 | 28004 | —— | —— | —— | —— | —— | 01 |
|  | 0 |  | —— | —— | GRSoe　　Ace | —— |  | —— | —— | 01 |
| ④ | 0 |  | 04 | 00080 | GRSoe　　Ace |  | —— | —— | —— | 01 |
|  | 0 |  | —— | —— | —— |  | —— | —— | —— | 01 |
| ⑤ | 0 |  | 11 | 6c705 | —— |  | —— | —— | —— | 01 |
|  | 0 |  | —— | —— | DATAoe　PSWce |  |  | —— | —— | 01 |
| ⑥ | 0 |  | 05 | 44006 | DATAoe　PSWce |  |  |  |  | 01 |
|  | 0 |  | —— | —— | GRSce　Soe |  |  |  |  | 01 |
| ⑦ | 0 |  | 06 | 20000 | GRSce　Soe |  |  |  |  | 01 |
|  | 0 |  | —— | —— | GRSoe |  |  |  |  | 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　1100B　。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 µAR= 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 µAR 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　CP1　（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　5　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 行修改为 ，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 I9~6 为 ，复位后重新执行，取目的操作数完成后， 微程序转移到地址 。

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 IR1~0 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

实验分工：单人完成

答：由于疫情返乡的原因，无法到实验室进行完整的实验，但是在对实验原理有一定程度的了解之后，加上由老师提供的实验数据的协助，虽然较为艰难，但仍然可以得出一些实验结论以及数据，总的来说实验不算困难，只要足够细心和耐心即可完成。