七、 实验七 多周期 CPU 实现

1 实验目的

1. 在单周期 CPU 实验完成的提前下，理解多周期的概念。
2. 熟悉并掌握多周期 CPU 的原理和设计。
3. 进一步提升运用 verilog 语言进行电路设计的能力。
4. 为后续实现流水线 cpu 的课程设计打下基础。

2 实验设备

1. 装有 Xilinx Vivado 的计算机一台。
2. LS-CPU-EXB-002 教学系统实验箱一套。

3 实验任务

1. 本次实验是对单周期 CPU 实验的拔高，也是为流水线 CPU 打下基础。前期的实验准备同单周期 CPU 的实验，在单周期 CPU 中只要求实现了十几条指令，但此处要求扩展到 30 多条指令。

多周期 CPU 是指，一条指令需要花费多个周期才能完成所有操作，在每个周期内只做一部分操作，比如：取指、译码、执行、访存、写回，此时，一条指令执行完，共需 5 个周期，每个周期只做一部分操作。

将 CPU 划分为多周期的优势在于，每个时钟周期内 CPU 需要做的工作就变少，因此频率可以更高，且每个部件做的事情单一了，比如取指部件只负责从指令存储器中取出指令，因此 CPU 可以进行流水工作，也相当于一个时钟周期完成一条指令。频率更高，依然相当于是一个周期完成一条指令，因此 CPU 可以运行的更快。

本次实验就是将实验六所实现的单周期 CPU 划分为多周期的，并扩展指令到 30 多条。

2. 依据单周期实验的设计框图，将其划分为多个功能块，每个功能块占用一个周期完成，即每个功能块从上一个功能块获取信息，作相关动作，完成后将结果锁存到寄存器中，作为下一个功能块的输入。建议划分为教课书上的 5 个功能块：取指、译码、执行、访存、写回，将理论与实践相结合。

3. 画出划分后的多周期 CPU 的框图，大致框图如图 8.1。从图中可以看出指令每个周期走完一个功能块，进入下一个功能块。标注的 clk 箭头是去往相邻模块的中间锁存器，是因为每个模块的输出需要锁存到寄存器中，下一个模块会从该寄存器中读出数据作为自己的输入，寄存器的锁存是需要时钟控制的。值得注意的是，写回模块所做的就是从访存模块获取要写入寄存器堆的数据和目的寄存器，送往寄存器堆，其所需的 clk 信号是最终发生在寄存器堆的写操作上的。自己设计的框图中要求力求精细，可参考教科书上的 5 级流水的框图。

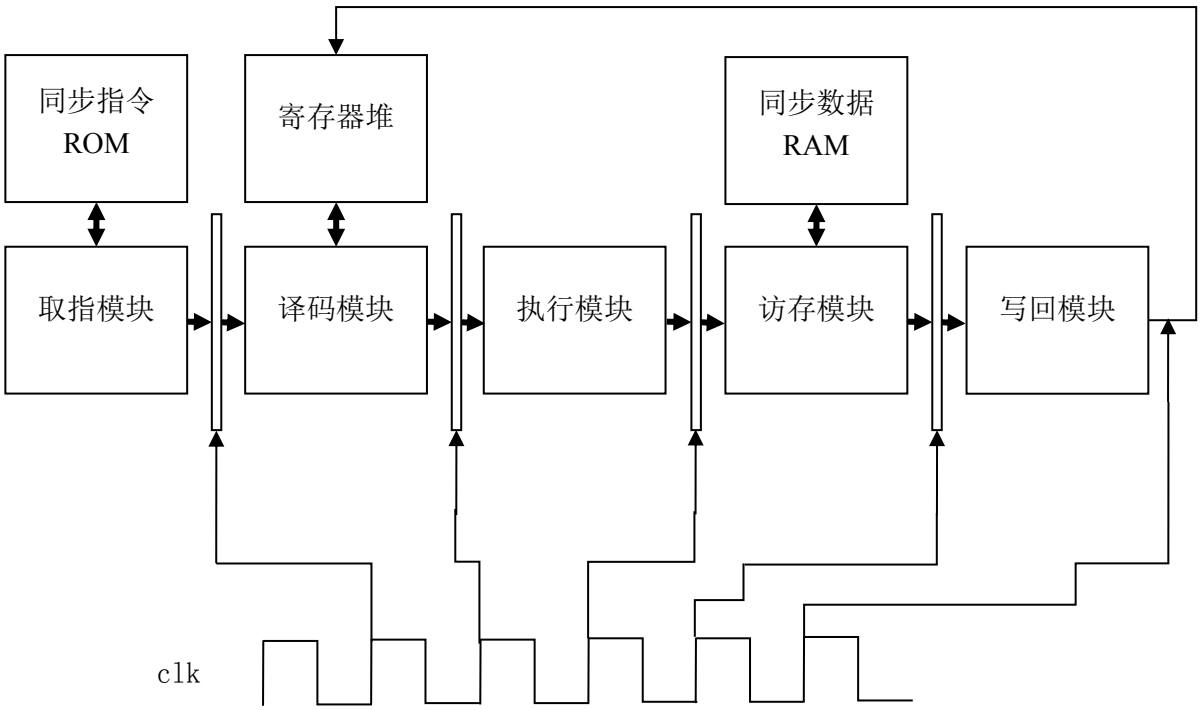

图 8.1 多周期 CPU 的大致框图

4. 本次实验是需要用到之前实验的结果的，比如 ALU 模块、寄存器堆模块、指令 ROM 模块和数据 RAM 模块，其中 ROM 和 RAM 建议使用调用库 IP 实例化的同步存储器，因为存储器在实际应用中基本都是同步读写的，为了更贴近真实情况，此处建议使用同步 RAM 和 ROM。

5. 在实验五中生成的同步 RAM 和 ROM，都是在发送地址后的下一拍才能获得对应数据的。故在在使用同步存储器时，从指令和数据存储器中读取数据就需要等待一拍时钟了，即取指令需要两拍时间，load 操作也需要两拍时间。在真实的处理器系统中，取指令和访存其实都是需要多拍时钟的。

6. 本次实验，同样需要完成表 8.1 和表 8.2 的填写。

表 8.1 mips 基础指令特性归纳表

指令类型	汇编指令	指令码	源操作数 1	源操作数 2	目的寄存器	功能描述
R 型指令	addu rd , rs , rt	000000_rs_|_rt_|rd_|00000|100001	[rs]	[rt]	rd	GPR[rd]=GPR[rs]+GPR[rt]
I 型指令	addiu rt,rs,imm	001001_rs_|_rt_|imm	[rs]	sign_ext(imm)	rt	GPR[rt]=GPR[rs]+sign_ext(imm)
J 型指令	j target	000010|target	PC	target		跳转, PC={PC[31:28],target,2'b00}

表 8.2 测试所用汇编程序详述

指令地址	汇编指令	结 果描 述	机器指令的机器码	
			16 进制	二进制
00H	addiu ,$1, $0,#1	[$1] = 0000_0001H	24010001	0010_0100_0000_0001__0000_0000_0000_0001

7. 根据设计的实验方案，使用 verilog 编写相应代码。

8. 对编写的代码进行仿真，得到正确的波形图。

9. 将以上设计作为一个单独的模块，设计一个外围模块去调用该模块，见图 8.2。外围模块中需调用封装好的 LCD 触摸屏模块，观察多周期 CPU 的内部状态，比如 32 个寄存器的值，各模块 PC 的值等。并且需要利用触摸功能输入特定数据 RAM 地址，从该 RAM 的调试端口读出数据显示在屏上，以达到实时观察数据存储器内部数据变化的效果。通过这些手段，可以在板上充分验证 CPU 的正确性。

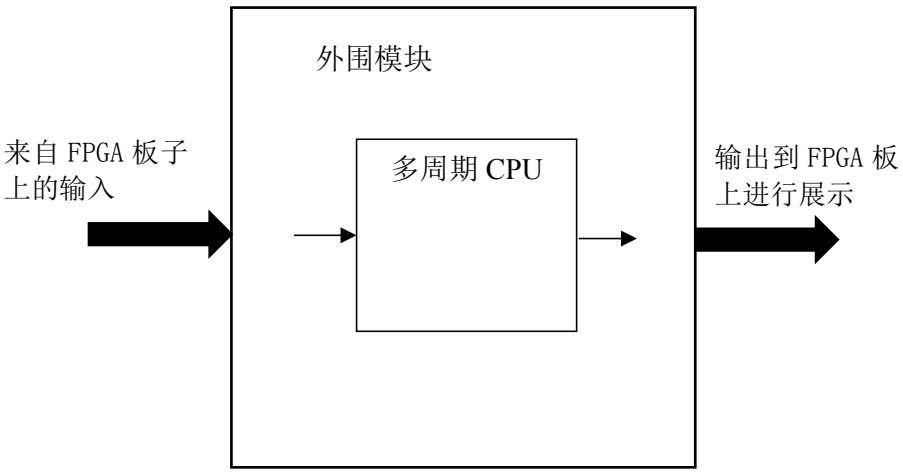

图 8.2 多周期 CPU 设计实验的顶层模块大致框图

10. 将编写的代码进行综合布局布线，并下载到实验箱中的 FPGA 板子上进行演示。

注意：

①：MIPS 架构中有延迟槽的设定，其本意是加快流水 CPU 的执行速度，故在多周期 CPU 中该设定无意义，反而带来了 CPU 实现上的麻烦，故建议在多周期 CPU 中不考虑延迟槽技术。

②：MIPS 架构中分支和跳转指令参与计算的 PC 值均为延迟槽指令对应的 PC(即分支跳转指令的 PC+4),而由于多周期不考虑延迟槽，故在实验中分支跳转指令参与计算使用本指令的 PC 值，故跳转的偏移量设置需要注意下。比如一条指令“beq,r0,r0,#2”在不考虑延迟槽的多周期 CPU 中，其跳转的目标地址为 beq 指令后

面的第 2 条。而在考虑延迟槽的流水 CPU 中，其跳转的目标地址为 beq 指令后面的第 3 条（即延迟槽指令后面的第 2 条）。这里需要理解清楚。

③：一般而言控制 CPU 运转的时钟是由 FPGA 板上的时钟输出提供的，但为了方便演示，我们需要在每一个时钟里查看一条指令的运算结果，故演示时理想的时钟是手动输入的，可以使用 FPGA 板上的脉冲开关代替时钟。

4 实验要求

1. 做好预习：

   1） 复习单周期 CPU 的实验内容，归纳常用的 MIPS 指令，确定自己准备实现的 MIPS 指令，对其进行分析，完成表 8.1 的填写；

   2） 依据自己设计中实现的指令，编写一段不少于 40 行的汇编程序，要求包含所有实现的指令，完成表 8.2 的填写；

   3） 认真学习多周期的概念，了解流水线的概念，明白划分为多周期的意义；

   4） 认真学习 CPU 各模块的功能，确认模块的划分。设计本次实验的方案，画出实验方案的设计框图，即补充完善图 8.1；

   5） 如果对 FPGA 板了解的话，可确定设计中与 FPGA 板上交互的接口，画出包含外围模块的整体设计框图，即补充完善图 8.2。

2. 实验实施：

   1） 确认多周期 CPU 的设计框图的正确性；

   2） 编写 verilog 代码，将表 8.2 中自己编写的汇编程序翻译为二进制，以 coe 文件的方式初始化到指令 ROM 中；

   3） 对该模块进行仿真，得出正确的波形，截图作为实验报告结果一项的材料，在仿真时需要将生成指令 ROM 时产生的.mif 文件拷贝到工程目录下，才能仿真成功；

   4） 完成调用多周期 CPU 的外围模块的设计，并编写代码；

   5） 对代码进行综合布局布线下载到实验箱里 FPGA 板上，进行上板验证。

3. 实验检查：

   1） 完成上板验证后，让指导老师或助教进行检查，进行现场演示。先解读表 8.2 中自己编写的汇编程序，然后采用手动输入时钟，每个周期查看 CPU 状态，按照检查人员的要求进行演示，检查指令运行结果的正确性，可对演示结果进行拍照作为实验报告结果一项的材料。

4. 实验报告的撰写：

   1） 实验结束后，需按照规定的格式完成实验报告的撰写。

5 参考设计

	andi rt,rs,imm	001100|rs|rt|imm	[rs]	zero_ext (imm)		rt	GPR[rt]=GPR[rs]&zero_ext (imm)
	lui rt,imm	001111|00000|rt|imm		{imm, 16'd0}		rt	GPR[rt]= {imm, 16'd0}
	ori rt,rs,imm	001101|rs|rt|imm	[rs]	zero_ext (imm)		rt	GPR[rt]=GPR[rs]|zero_ext (imm)

表 8.4 多周期 CPU 实现的 mips 指令特性归纳（续）

指令类型	汇编指令	指令码	源操作数 1	源操作数 2	源操作数 3	目的寄存器	功能描述
I 型指令	xori rt,rs,imm	001110|rs|rt|imm	[rs]	zero_ext (imm)		rt	GPR[rt]=GPR[rs]^zero_ext (imm)
J 型指令	j target	000010|target					PC={PC[31:28],target<<2}
	jal target	000011|target					GPR[31]=PC,PC={PC[31:28],target<<2}

多周期 CPU 设计在单周期 CPU 基础上，主要做两部分改进。第一部分是控制单元，增加控制电路使每一个时钟只有一个阶段的电路产生的结果有效，并锁存上一阶段的结果用于后续阶段的运行；第二部分是数据通路，增加实现新增指令的电路。

第一部分的改进主要是增加状态机控制及增加各阶段之间的用于锁存的寄存器。由于有 5 个阶段，状态机共有 6 个状态：空闲（IDLE）、取指（FETCH）、译码（DECODE）、执行（EXE）、访存（MEM）、写回（WB），如下图：

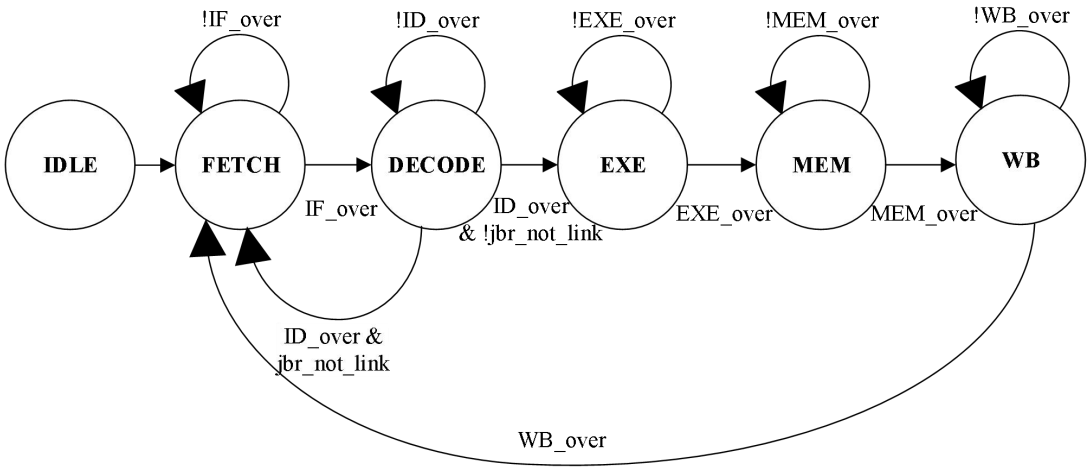

图 8.3 多周期 CPU 的状态机

空闲（IDLE）状态：CPU 在复位时，所有阶段电路都无效，CPU 等待复位结束开始下一状态——取指。

取指（FETCH）状态：在状态机进入取指状态的同时，PC 更新为下一 PC 值。故取指状态下，将 PC 值作为指令存储器的地址去取指令。由于同步指令存储器在下一时钟周期返回指令，因此取指需要两个时钟周期的时间。当取指结束后锁存取指阶段产生的结果——当前 PC 值和指令。

译码（DECODE）状态：类似于单周期 CPU 的译码阶段，主要完成指令译码、读寄存器、判断跳转等。控制单元区分各条指令并产生用于译码、执行、访存、写回的控制信号。当译码结束后锁存译码阶段产生的结果用于下一状态执行：分别用于执行、

访存、写回的控制信号、用于执行阶段的两个源操作数、用于访存阶段的写入内存数据、用于写回阶段的写寄存器地址。

执行（EXE）状态：ALU 模块完成操作。当执行结束后锁存执行阶段产生的结果及前级传递的结果：用于访存、写回的控制信号、ALU 结果、内存写入数据、寄存器写地址。

访存（MEM）状态：完成对数据存储器的读或写，并选择出将要写回寄存器的值。当访存结束后锁存访存阶段产生的结果及前级传递的结果：用于写回的控制信号、写回数据、写回地址。

写回（WB）状态：完成寄存器写入。

CPU 复位结束，状态机由 IDLE 进入取指状态，其后在每次上一级结束信号有效的时进入下一状态，写回级结束后返回取指级。当然也有例外，当指令是跳转而非链接跳转指令时，在译码状态后直接返回取下一指令不需要经过执行等后续阶段。

多周期 CPU 的实现框图如下：

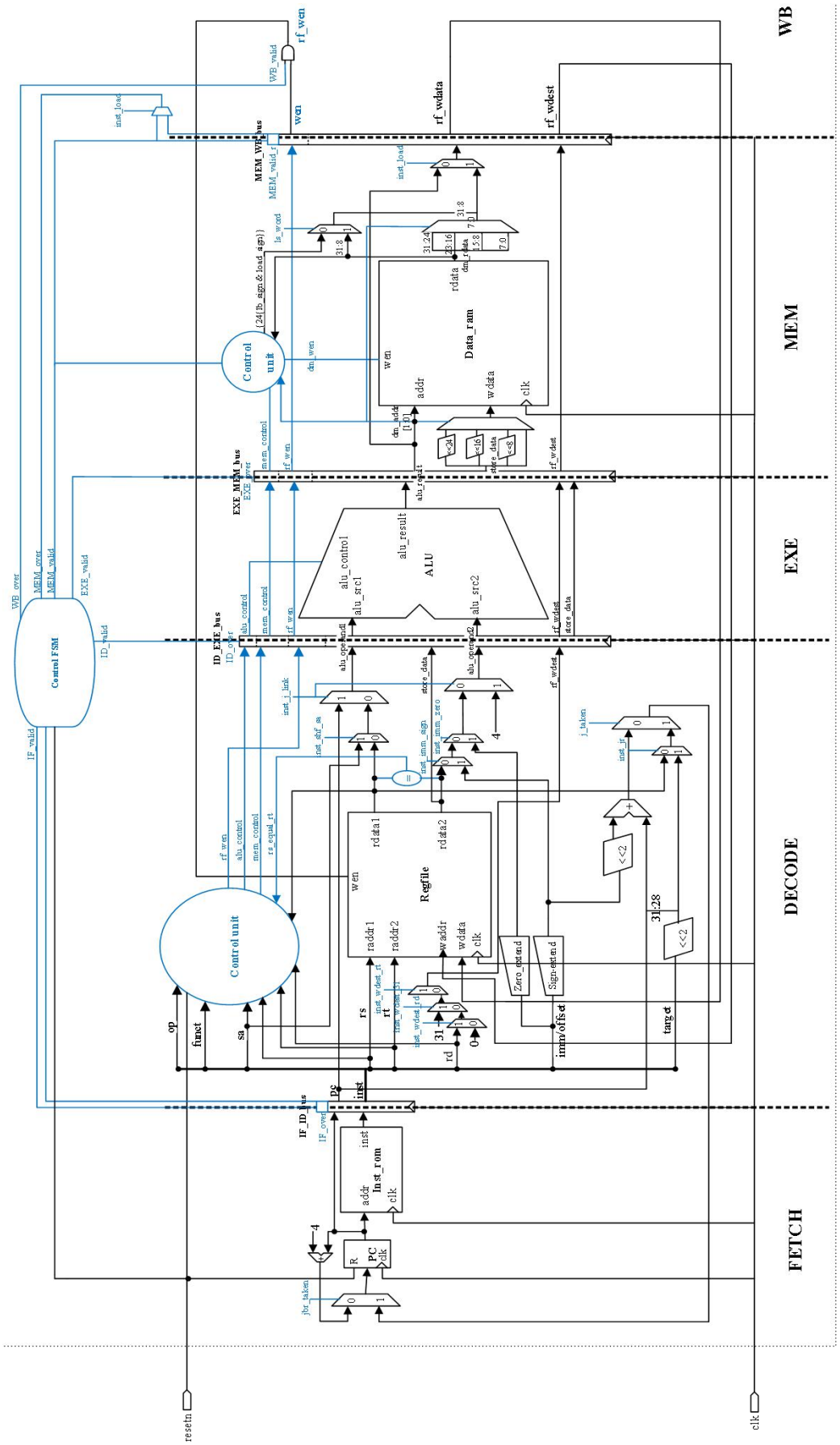

图 8.4 多周期 CPU 的实现框图

表 8.5 多周期 CPU 测试所用汇编程序详述（续）

指令地址	汇编指令	结果描述	机器指令的机器码	
			16 进制	二进制
84H	addiu $11,$0,#140	[$11] = 0000_008CH	240B008C	0010_0100_0000_1011_0000_0000_1000_1100
88H	bltz $16, #6	跳转到 A0H	06000006	0000_0110_0000_0000_0000_0000_0000_0110
8CH	lw $28,#3($10)	[$28] = 000C_000DH /000C_880DH	8D5C0003	1000_1101_0101_1100_0000_0000_0000_0011
90H	bne $28,$29,#7	不跳转/跳转 ACH	179D0007	0001_0111_1001_1101_0000_0000_0000_0111
94H	sb $15,#8($5)	Mem[0000_0015H] = 0000_0088H	A0AF0008	1010_0000_1010_1111_0000_0000_0000_1000
98H	lb $18,#8($5)	[$18] =FFFF_FF88H	80B20008	1000_0000_1011_0010_0000_0000_0000_1000
9CH	lbu $19,#8($5)	[$19] = 0000_0088H	90B30008	1001_0000_1011_0011_0000_0000_0000_1000
A0H	sltiu $24,$15,#0xFFFF	[$24] = 0000_0001H	2DF8FFFF	0010_1101_1111_1000_1111_1111_1111_1111
A4H	or $29,$12,$5	[$29] = 000C000DH	0185E825	0000_0001_1000_0101_1110_1000_0010_0101
A8H	jr $11	跳转指令 8CH	01600008	0000_0001_0110_0000_0000_0000_0000_1000
ACH	andi $20,$15,#0xFFFF	[$20] = 0000_FF88H	31F4FFFF	0011_0001_1111_0100_1111_1111_1111_1111
B0H	ori $21,$15,#0xFFFF	[$21] =FFFF_FFFFH	35F5FFFF	0011_0101_1111_0101_1111_1111_1111_1111
B4H	xori $22,$15,#0xFFFF	[$22] = FFFF_0077H	39F6FFFF	0011_1001_1111_0110_1111_1111_1111_1111
B8H	j #00H	跳转指令 00H	08000000	0000_1000_0000_0000_0000_0000_0000_0000

5.2 实验顶层模块框图

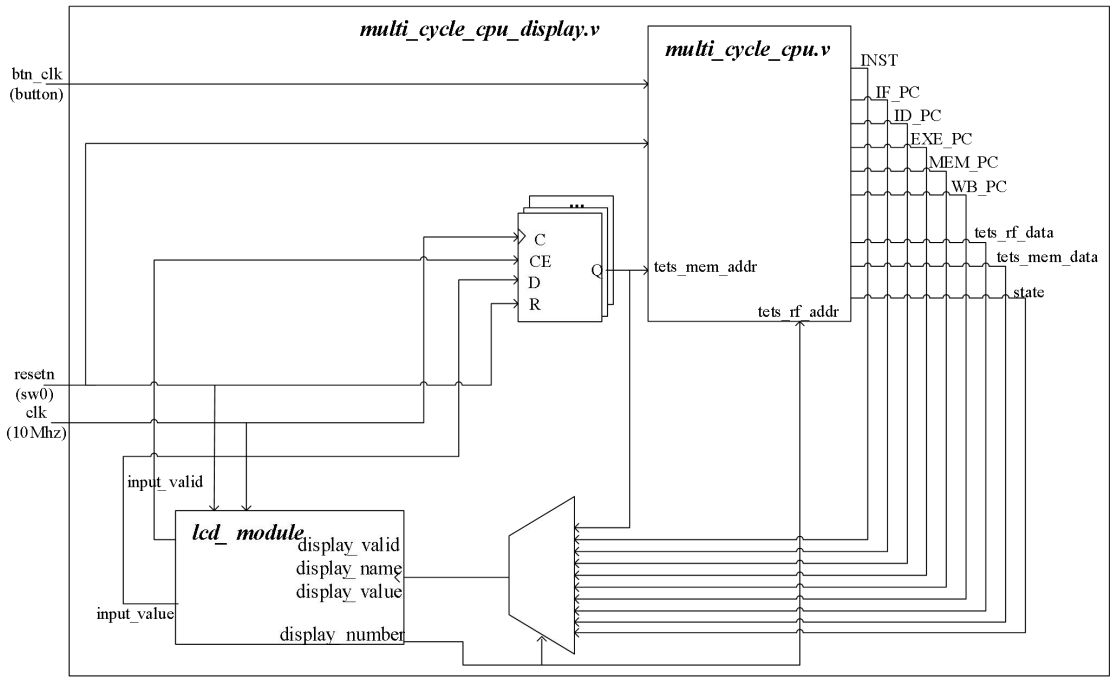

图 8.5 多周期 CPU 参考设计的顶层模块框图

5.3 源代码

multi_cycle_cpu_display.v 源代码：

```
1  `timescale 1ns / 1ps
2  //*************************************************************
   *
```