目录

（一）概述

MIPS 是一种精简指令集（RISC）架构。本文旨在通过设计一个基于 MIPS 指令系统的 RISC 处理器，来深刻理解处理器结构和计算机系统的整体工作原理。下文中的处理器指令字长为 32 位，包含 32 个 32 位通用寄存器 R_0~R_{31}，能够完成 10 条机器指令。

（二）指令格式设计

本实验分别从 MIPS 指令系统的 R 型指令中选取“ADD”、“SUB”、“AND”、“OR”、“SLL”这 5 条指令，从 I 型指令中选取“ADDI”、“LW”、“BEQ”、“SW”这 4 条指令，从 J 型指令中选取“J”这 1 条指令进行处理器设计。

2.1 R 型指令

R 型指令基本指令格式如图 1 所示：

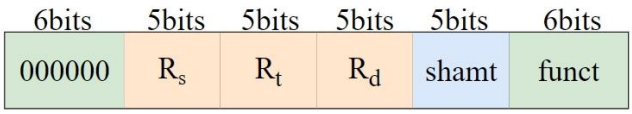

图 1 R 型指令基本指令格式

（1）ADD 指令

ADD 指令格式如图 2 所示：

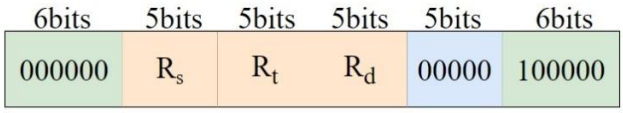

图 2 ADD 指令格式

ADD 指令的汇编格式为 add rd, rs, rt，完成 R[rd]←R[rs]+R[rt]，即将寄存器 rs 中的数与寄存器 rt 中的数相加，并将结果保存在寄存器 rd 中。

（2）SUB 指令

SUB 指令格式如图 3 所示：

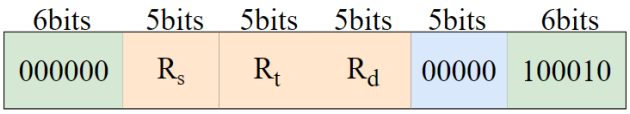

图 3 SUB 指令格式

SUB 指令的汇编格式为 sub rd, rs, rt，完成 R[rd]←R[rs]-R[rt]，即将寄存器 rs 中的数与寄存器 rt 中的数相减，并将结果保存在寄存器 rd 中。

（3）AND 指令

AND 指令格式如图 4 所示：

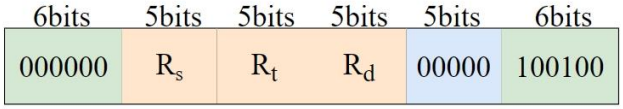

图 4 AND 指令格式

AND 指令的汇编格式为 and rd, rs, rt，完成 R[rd]←R[rs]&R[rt]，即将寄存器 rs 中的数与寄存器 rt 中的数进行“与”运算，并将结果保存在寄存器 rd 中。

（4）OR 指令

OR 指令格式如图 5 所示：

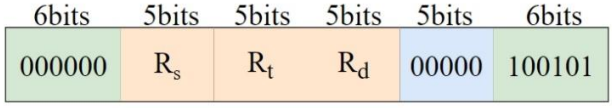

图 5 OR 指令格式

OR 指令的汇编格式为 or rd, rs, rt，完成 R[rd]←R[rs]|R[rt]，即将寄存器 rs 中的数与寄存器 rt 中的数进行“或”运算，并将结果保存在寄存器 rd 中。

（5）SLL 指令

SLL 指令格式如图 6 所示：

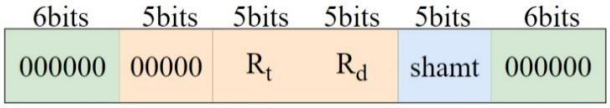

图 6 SLL 指令格式

SLL 指令的汇编格式为 sll rd, rt, shamt，完成 R[rd]←R[rt]<<shamt，即将寄存器 rt 中的数左移 shamt 位，并将结果保存在寄存器 rd 中。

2.2 I 型指令

I 型指令基本指令格式如图 7 所示：

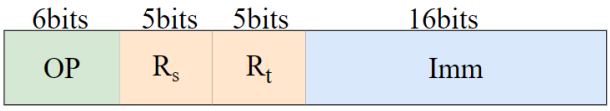

图 7 I 型指令基本指令格式

（1）ADDI 指令

ADDI 指令格式如图 8 所示：

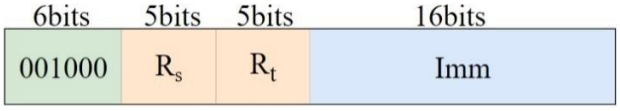

图 8 ADDI 指令格式

ADDI 指令的汇编格式为 addi rt, rs, imm，完成 R[rt]←R[rs]+imm，即将寄存器 rs 中的数与立即数 imm 相加，并将结果保存在寄存器 rt 中。

（2）LW 指令

LW 指令格式如图 9 所示：

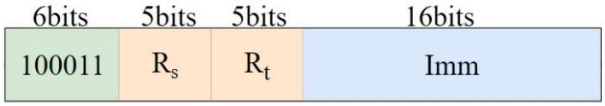

图 9 LW 指令格式

LW 指令的汇编格式为 lw rt, imm(rs)，完成 R[rt]←M[R[rs]+imm]，即将寄存器 rs 中的地址与立即数 imm 相加获得访存地址，从该地址中取数，并将结果保存在寄存器 rt 中。

（3）BEQ 指令

BEQ 指令格式如图 10 所示：

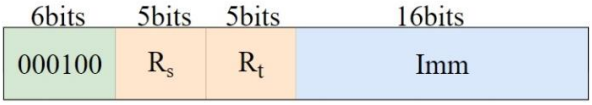

图 10 BEQ 指令格式

BEQ 指令的汇编格式为 beq rs, rt, imm，完成 if(R[rs]==R[rt])　PC=PC+4+imm<<2，即先判断寄存器 rs 中的数与寄存器 rt 中的数是否相等，如果相等，则跳转至 PC+4+imm<<2 的位置。

（4）SW 指令

SW 指令格式如图 11 所示：

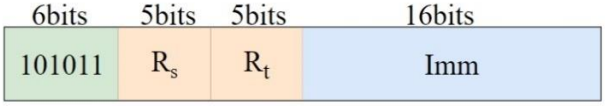

图 11 SW 指令格式

SW 指令的汇编格式为 sw rt, imm(rs)，完成 M[R[rs]+imm]←R[rt]，即将寄存器 rs 中的地址与立即数 imm 相加获得访存地址，并将寄存器 rt 的数写回到该地址。

2.3 J 型指令

J 型指令基本指令格式如图 12 所示：

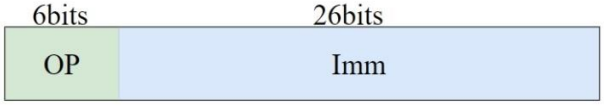

图 12 J 型指令基本指令格式

（1）J 指令

J 指令格式如图 13 所示：

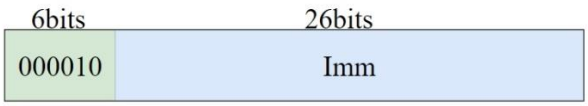

图 13 J 指令格式

J 指令的汇编格式为 j imm，完成 PC=(PC+4)||imm||00，即将 PC+4 与立即数 imm 做连接运算，末尾补两个 0 得到跳转地址，并无条件跳转到该地址。

（三）微操作定义

将处理器工作流程分为取指、译码、运算、访存、写回五个阶段进行。

3.1 ADD 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)+(R(IR[16~20])→Z	Y 的值与寄存器 Rt 的值相加的结果→Z

（4）写回阶段

微操作	微操作描述
Z→R(IR[11~15])	Z 的值→寄存器 R_d

3.2 SUB 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读

M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	**微操作描述**
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	**微操作描述**
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)-(R(IR[16~20]))→Z	Y 的值与寄存器 Rt 的值相减的结果→Z

（4）写回阶段

微操作	**微操作描述**
Z→R(IR[11~15])	Z 的值→寄存器 R_d

3.3 AND 指令

（1）取指阶段

微操作	**微操作描述**
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	**微操作描述**
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	**微操作描述**
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)&(R(IR[16~20]))→Z	Y 的值与寄存器 Rt 的值进行“与”运算的结果→Z

（4）写回阶段

微操作	**微操作描述**
Z→R(IR[11~15])	Z 的值→寄存器 R_d

3.4 OR 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)|(R(IR[16~20]))→Z	Y 的值与寄存器 Rt 的值进行“或”运算的结果→Z

（4）写回阶段

微操作	微操作描述
Z→R(IR[11~15])	Z 的值→寄存器 R_d

3.5 SLL 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[16~20])→Y	寄存器 R_t 的值→Y
(Y)<< (R(IR[6~10]))→Z	Y 的值左移 shamt 位的结果→Z

（4）写回阶段

微操作	微操作描述
Z→R(IR[11~15])	Z 的值→寄存器 R_d

3.6 ADDI 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)+ (R(IR[0~15]))→Z	Y 的值与立即数 imm 相加的结果→Z

（4）写回阶段

微操作	微操作描述
Z→R(IR[16~20])	Z 的值→寄存器 R_t

3.7 LW 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)+ (R(IR[0~15]))→Z	Y 的值与立即数 imm 相加的结果→Z

（4）访存阶段

微操作	微操作描述
Z→MAR	访存地址→MAR
1→R	命令存储器读
M(MAR)→MDR	访存地址中的数据从存储器中读至 MDR

（5）写回阶段

微操作	微操作描述
MDR→R(IR[16~20])	MDR 的值→寄存器 R_t

3.8 BEQ 指令

（1）取指阶段

微操作	微操作描述
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	微操作描述
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	微操作描述
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)==(R(IR[16~20]))→ALU	计算 Y 的值与寄存器 R_t 的值是否相等
(PC)+(R(IR[0~15]))<<2→Z	跳转地址→Z

（4）访存阶段

微操作	微操作描述
Z→PC	Z 的值→PC

3.9 SW 指令

（1）取指阶段

微操作	微操作描述

PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	**微操作描述**
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	**微操作描述**
R(IR[21~25])→Y	寄存器 R_s 的值→Y
(Y)+ (R(IR[0~15]))→Z	Y 的值与立即数 imm 相加的结果→Z

（4）访存阶段

微操作	**微操作描述**
Z→MAR	访存地址→MAR
1→W	命令存储器写
R(IR[16~20])→MDR	寄存器 R_t 中的数据读至 MDR
MDR→M(MAR)	数据写至存储器中

3.10 J 指令

（1）取指阶段

微操作	**微操作描述**
PC→MAR	现行指令地址→MAR
1→R	命令存储器读
M(MAR)→MDR	现行指令从存储器中读至 MDR
MDR→IR	现行指令→IR
(PC)+4→PC	形成下一条指令的地址

（2）译码阶段

微操作	**微操作描述**
OP(IR)→CU	指令的操作码→CU 译码

（3）运算阶段

微操作	**微操作描述**
(PC[28~31])||(R(IR[0~25]))||00→Z	跳转地址→Z

（4）访存阶段

微操作	**微操作描述**

Z→PC	Z 的值→PC

（四）节拍划分

4.1 取指阶段

节拍	微指令
T0	PC→MAR 1→R
T1	Ad(CMDR)→CMAR
T2	M(MAR)→MDR (PC+4)→PC
T3	Ad(CMDR)→CMAR
T4	MDR→IR

4.2 译码阶段

节拍	微指令
T0	OP(IR)→微地址形成部件
T1	OP(IR)→微地址形成部件→CMAR

4.3 运算阶段

（1）ADD 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)+(R(IR[16~20])→Z
T3	Ad(CMDR)→CMAR

（2）SUB 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)-(R(IR[16~20])→Z
T3	Ad(CMDR)→CMAR

（3）AND 指令

节拍	微指令

T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)&(R(IR[16~20])→Z
T3	Ad(CMDR)→CMAR

（4）OR 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)|(R(IR[16~20])→Z
T3	Ad(CMDR)→CMAR

（5）SLL 指令

节拍	微指令
T0	R(IR[16~20])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)<< (R(IR[6~10]))→Z
T3	Ad(CMDR)→CMAR

（6）ADDI 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)+ (R(IR[0~15]))→Z
T3	Ad(CMDR)→CMAR

（7）LW 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)+ (R(IR[0~15]))→Z
T3	Ad(CMDR)→CMAR

（8）BEQ 指令

节拍	微指令
T0	R(IR[21~25])→Y (PC)+(R(IR[0~15]))<<2→Z
T1	Ad(CMDR)→CMAR
T2	(Y)==(R(IR[16~20]))→ALU
T3	Ad(CMDR)→CMAR

（9）SW 指令

节拍	微指令
T0	R(IR[21~25])→Y
T1	Ad(CMDR)→CMAR
T2	(Y)+ (R(IR[0~15]))→Z
T3	Ad(CMDR)→CMAR

（10）J 指令

节拍	微指令
T0	(PC[28~31])‖(R(IR[0~25]))‖00→Z
T1	Ad(CMDR)→CMAR

4.4 访存阶段

（1）LW 指令

节拍	微指令
T0	Z→MAR 1→R
T1	Ad(CMDR)→CMAR
T2	M(MAR)→MDR
T3	Ad(CMDR)→CMAR

（2）BEQ 指令

节拍	微指令
T0	Z→PC
T1	Ad(CMDR)→CMAR

（2）SW 指令

节拍	微指令
T0	Z→MAR 1→W
T1	Ad(CMDR)→CMAR
T2	R(IR[16~20])→MDR
T3	Ad(CMDR)→CMAR
T4	MDR→M(MAR)
T5	Ad(CMDR)→CMAR

（3）J 指令

节拍	微指令
T0	Z→PC
T1	Ad(CMDR)→CMAR

4.5 写回阶段

（1）ADD 指令

节拍	微指令
T0	Z→R(IR[11~15])
T1	Ad(CMDR)→CMAR

（2）SUB 指令

节拍	微指令
T0	Z→R(IR[11~15])
T1	Ad(CMDR)→CMAR

（3）AND 指令

节拍	微指令
T0	Z→R(IR[11~15])
T1	Ad(CMDR)→CMAR

（4）OR 指令

节拍	微指令
T0	Z→R(IR[11~15])
T1	Ad(CMDR)→CMAR

（5）SLL 指令

节拍	微指令
T0	Z→R(IR[11~15])
T1	Ad(CMDR)→CMAR

（6）ADDI 指令

节拍	微指令
T0	Z→R(IR[16~20])
T1	Ad(CMDR)→CMAR

（7）LW 指令

节拍	微指令
T0	MDR→R(IR[16~20])
T1	Ad(CMDR)→CMAR

（五）处理器设计

5.1 处理器结构概述

采用变长指令周期的设计思想，通过多周期实现，以此设计的处理器结构框图如图 14

所示。该处理器能够完成“ADD”、“LW”、“J”等 10 条指令的取指、译码、运算、访存、写回五个阶段的工作。

程序计数器 PC、指令寄存器 IR、数据寄存器 DR、通用寄存器 Y、通用寄存器 Z、存储器、控制器以及寄存器堆都受时钟信号控制。控制器发出控制信号，控制多路选择器进行数据选择，控制 ALU 进行不同的运算，控制存储器和寄存器堆的读写，控制是否进行跳转等。

其中，有关控制器的具体控制信号描述如下：

（1）Load

控制访存的地址。“0”表示“PC→MAR”，“1”表示“Z→MAR”。

（2）MemR

控制存储器读。

（3）MemW

控制存储器写。

（4）RegDst

控制多路选择器，选择写回的寄存器。“0”表示写回到寄存器 R_t，“1”表示写回到寄存器 R_d。

（5）WritetoReg

控制多路选择器，选择写回到寄存器的数据。“0”表示该数据来自存储器，“1”表示数据来自通用寄存器 Z。

（6）RegR

控制寄存器堆读。

（7）RegW

控制寄存器堆写。

（8）RegtoY

控制多路选择器，选择写入通用寄存器 Y 的数据。“0”表示该数据来自寄存器 R_s，“1”表示该数据来自寄存器 R_t。

（9）ALUSrcA

控制多路选择器，选择进行运算的数据。“0”表示数据来自 PC，“1”表示数据来自通用寄存器 Y。

（10）ALUSrcB

控制多路选择器，选择进行运算的数据。“000”表示数据为 4（完成 PC+4），“001”表示数据来自寄存器 R_t，“010”表示数据为 R 型指令的 shamt 部分，“011”表示数据为 I 型指令的立即数 Imm 部分进行扩展的结果，“100”表示数据为 I 型指令的立即数 Imm 部分进行扩展再左移两位的结果，“101”表示数据为 J 型指令的立即数 Imm 部分。

（11）ALUOP

控制 ALU 进行不同的运算。包括加法、减法、“与”运算、“或”运算、逻辑左移、连接运算等。

（12）PCSrc

控制多路选择器，控制 PC 的数据来源。“0”表示数据来自 ALU 的直接运算结果（PC+4→PC），“1”表示数据来自通用寄存器 Z（Z→PC）。

（13）Branch

控制条件转移分支。

（14）PCWrite

控制 PC 写。

（15）IRWrite

控制 IR 写

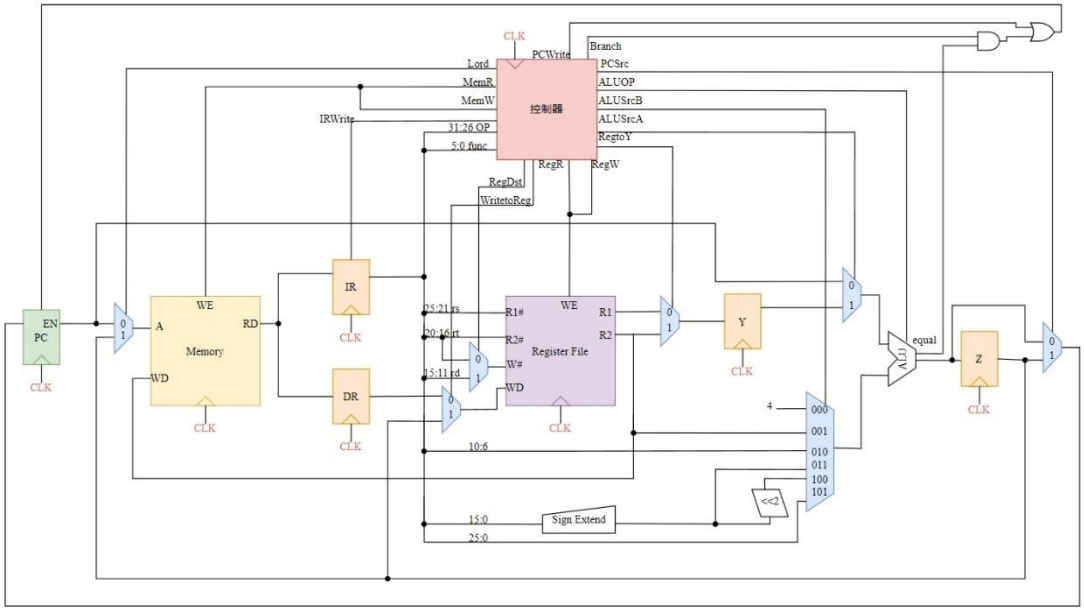

图 14 处理器结构设计框图

5.2 处理器功能描述

该处理器在时钟信号控制下，可以以多周期实现 10 条指令从取指到执行的全过程。由于可将 CMDR 的下地址字段 Ad(CMDR)直接接到控制存储器的地址线上，并由下一个时钟周期的上升沿将该地址单元的内容（微指令）读到 CMDR 中，便能做到在一个时钟周期内读出并执行一条微指令。下面分别以“ADD”、“LW”、“J”3 条指令为例，说明处理器在每个阶段的运行过程，以此说明处理器在每个阶段的功能。

5.2.1 取指阶段

取指阶段的数据通路如图 15 所示：

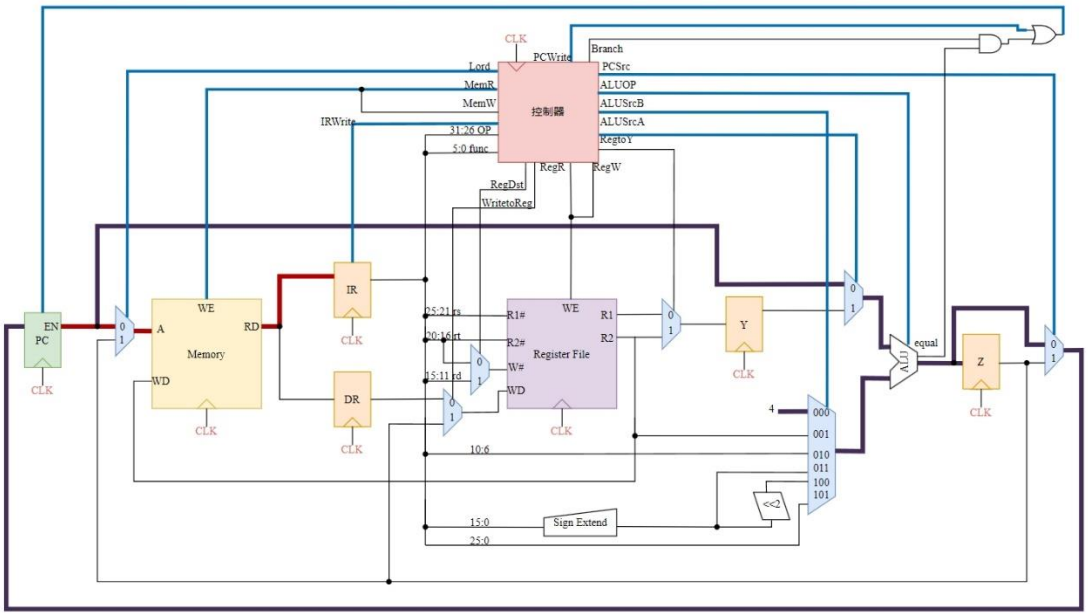

图 15 取指阶段的数据通路

T0 节拍，控制器发出 Lord 信号，完成 PC→MAR；

T1 节拍，控制器发出 MemR 信号，完成 M(MAR)→MDR；控制器发出 ALUSrcA 信号，选择第 0 路（即 PC），控制器发出 ALUSrcB 信号，选择第 0 路（即 4），在 ALU 中完成加法运算，控制器发出 PCSrc 信号，选择第 0 路，将值送回 PC，控制器发出 PCWrite 信号，将值写回 PC，完成 PC+4→PC；

T2 节拍，控制器发出 IRWrite 信号，完成 MDR→IR。

5.2.2 译码阶段

译码阶段的数据通路如图 16 所示：

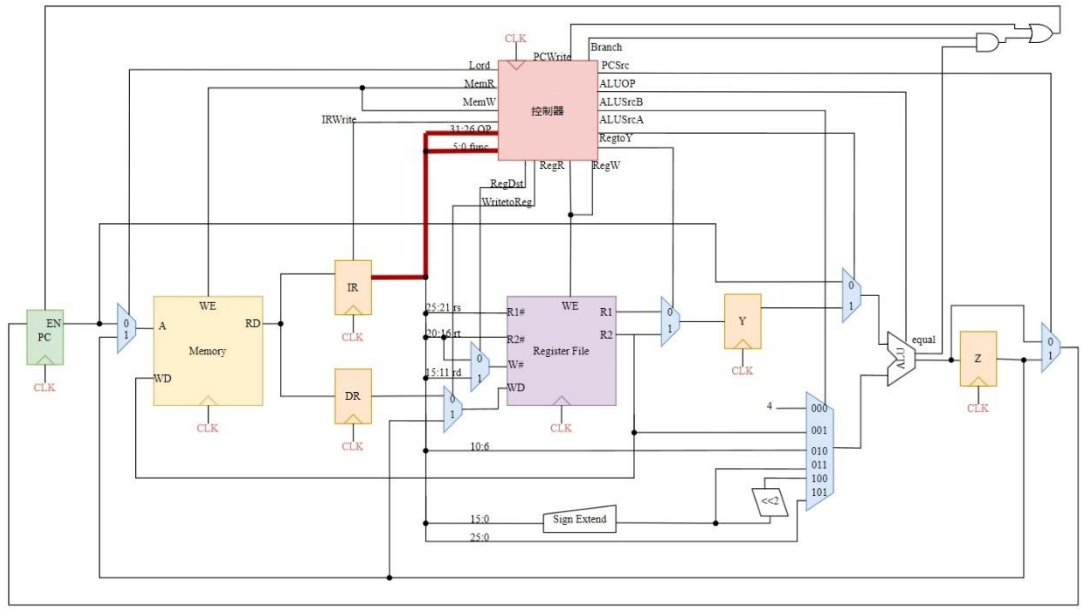

图 16 译码阶段的数据通路

T0 节拍，完成 OP(IR)→微地址形成部件。

5.2.3 后续阶段——ADD 指令

（1）运算阶段

运算阶段的数据通路如图 17 所示：

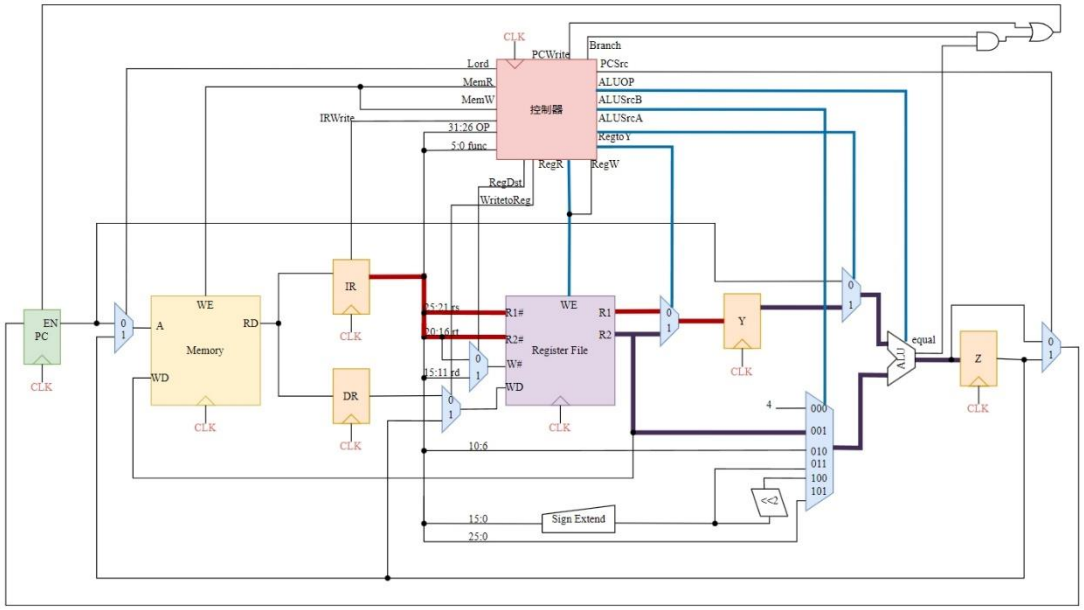

图 17 ADD 指令运算阶段的数据通路

T0 节拍，控制器发出 RegR 信号，读出寄存器 R_s 的值，控制器发出 RegtoY 信号，将该值写入通用寄存器 Y，完成 R(IR[21~25])→Y；

T1 节拍，控制器发出 ALUSrcA 信号，选择第 1 路（即 Y），控制器发出 ALUSrcB 信号，选择第 1 路（即 R_t），控制器发出 ALUOP 信号，在 ALU 中完成加法运算，将结果送至通用寄存器 Z，完成(Y)+(R(IR[16~20])→Z。

（2）写回阶段

写回阶段的数据通路如图 18 所示：

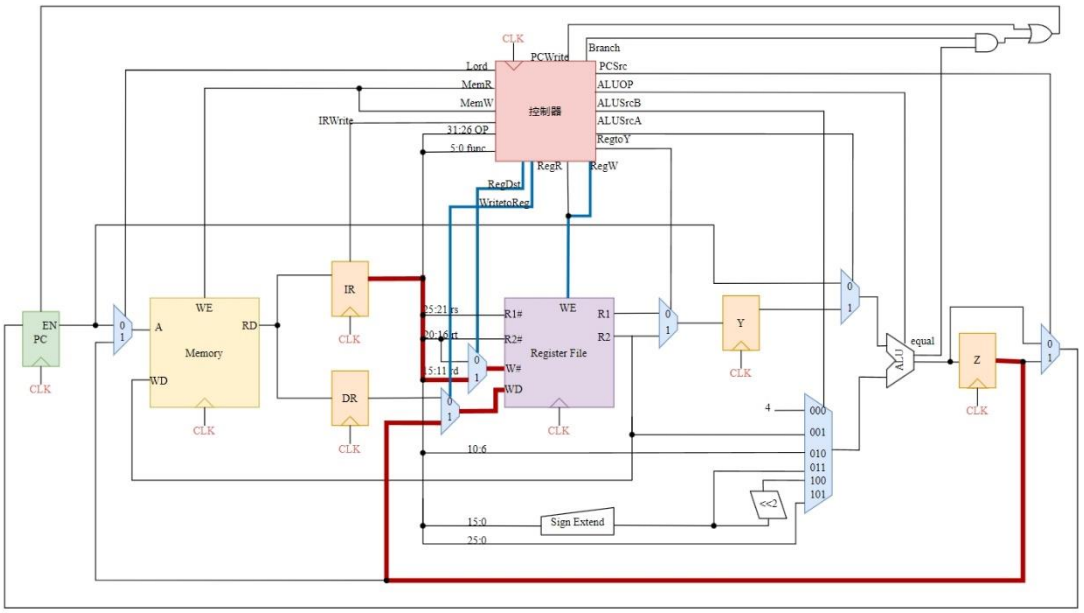

图 18 ADD 指令写回阶段的数据通路

T0 节拍，控制器发出 RegDst 信号，选择第 1 路（即寄存器 R_d），控制器发出 WritetoReg 信号，选择第 1 路（即 Z），控制器发出 RegW 信号，完成 Z→R(IR[11~15])。

5.2.4 后续阶段——LW 指令

（1）运算阶段

运算阶段的数据通路如图 19 所示：

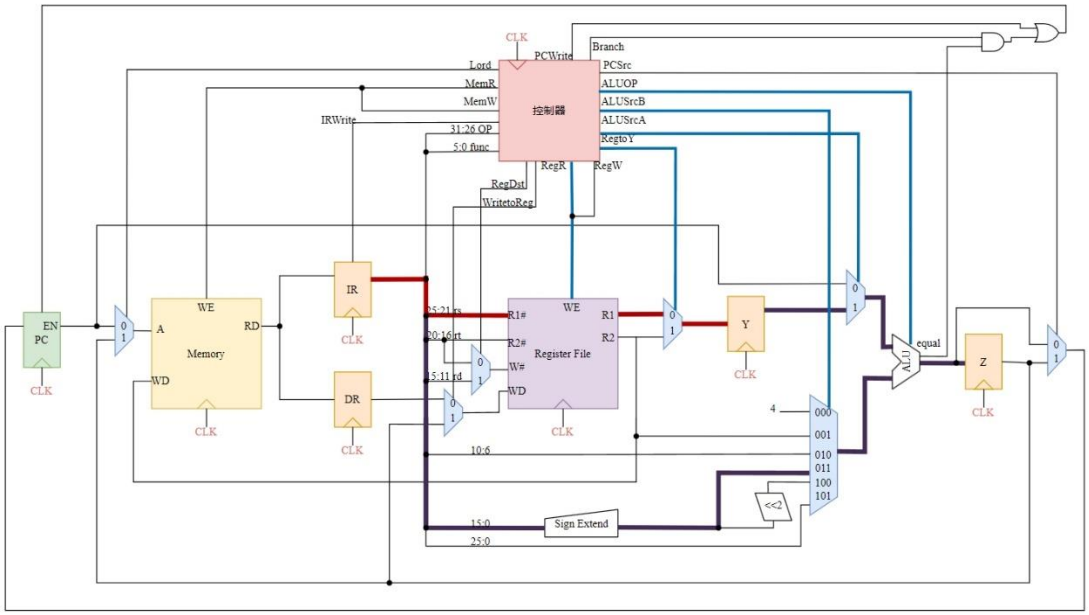

图 19 LW 指令运算阶段的数据通路

T0 节拍，控制器发出 RegR 信号，读出寄存器 R_s 的值，控制器发出 RegtoY 信号，将该值写入通用寄存器 Y，完成 R(IR[21~25])→Y；

T1 节拍，控制器发出 ALUSrcA 信号，选择第 1 路（即 Y），控制器发出 ALUSrcB 信号，选择第 4 路（即扩展后的立即数），控制器发出 ALUOP 信号，在 ALU 中完成加法运算，将结果送至通用寄存器 Z，完成(Y)+(R(IR[0~15])→Z。

（2）访存阶段

访存阶段的数据通路如图 20 所示：

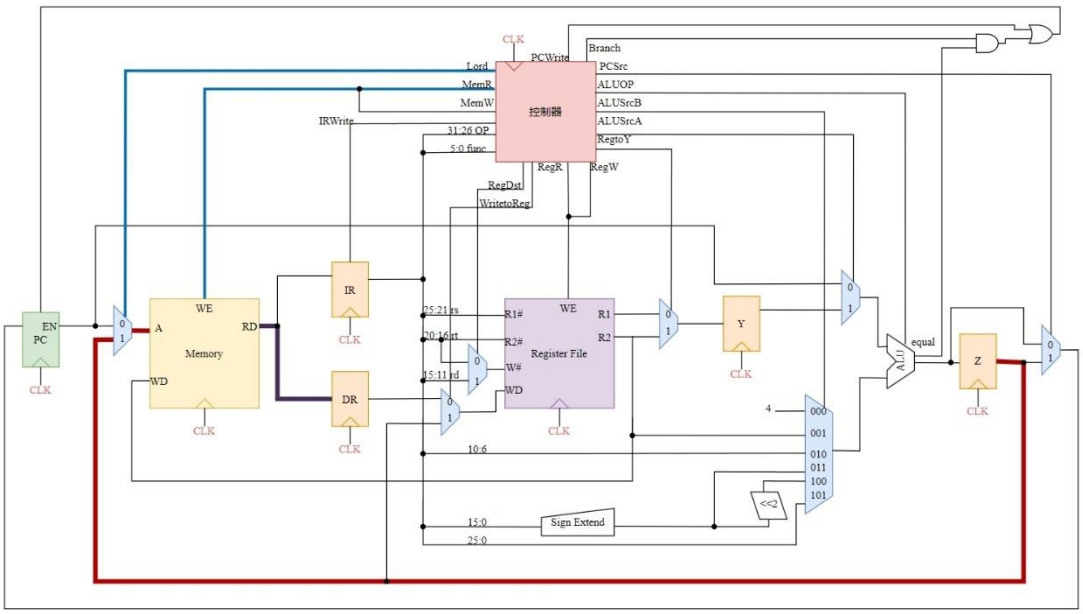

图 20 LW 指令访存阶段的数据通路

T0 节拍，控制器发出 Load 信号，选择第 1 路（即 Z），完成 Z→MAR；

T1 节拍，控制器发出 MemR 信号，完成 M(MAR)→MDR。

（3）写回阶段

写回阶段的数据通路如图 21 所示：

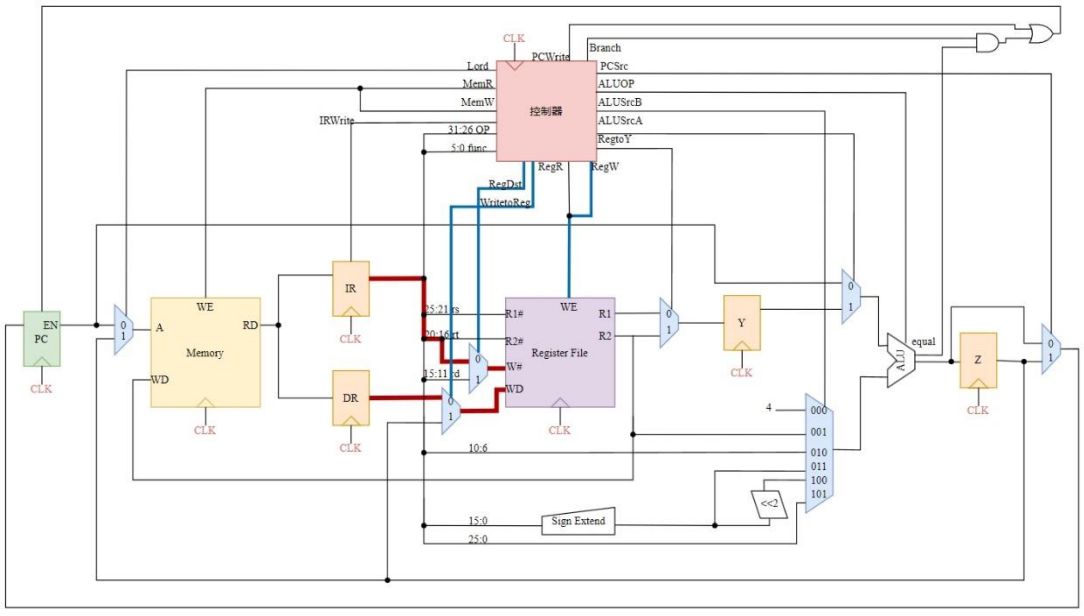

图 21 LW 指令写回阶段的数据通路

T0 节拍，控制器发出 RegDst 信号，选择第 0 路（即寄存器 R_t），控制器发出 WritetoReg 信号，选择第 0 路（即 MDR），控制器发出 RegW 信号，完成 MDR→R(IR[16~20])。

5.2.5 后续阶段——J 指令

（1）运算阶段

运算阶段的数据通路如图 22 所示：

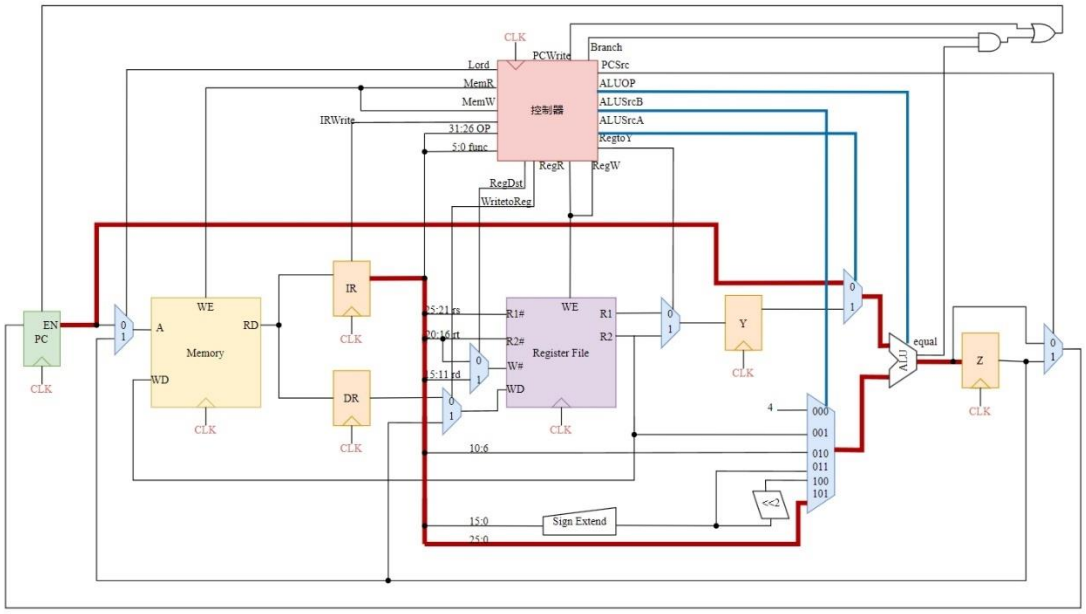

图 22 J 指令运算阶段的数据通路

T0 节拍，控制器发出 ALUSrcA 信号，选择 0 路（即 PC），控制器发出 ALUSrcB 信号，选择第 6 路（即立即数），控制器发出 ALUOP 信号，在 ALU 中完成连接运算，将结果送至通用寄存器 Z，完成(PC[28~31])||(R(IR[0~25]))||00→Z。

（2）访存阶段

访存阶段的数据通路如图 23 所示：

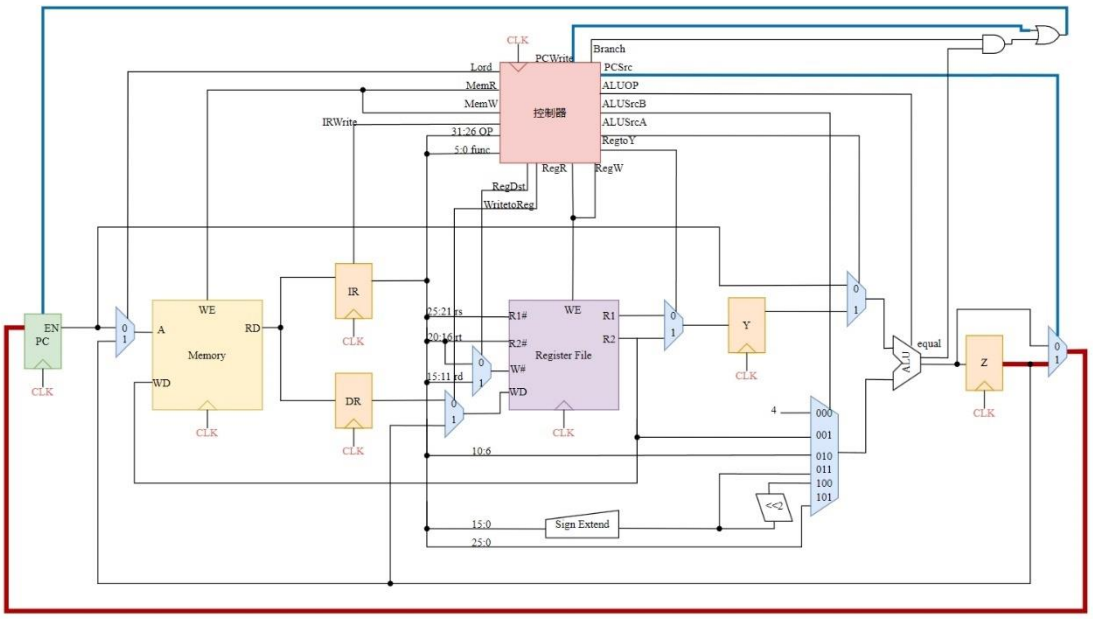

图 23 J 指令访存阶段的数据通路

T0 节拍，控制器发出 PCSrc 信号，选择第 1 路（即 Z），控制器发出 PCWrite 信号，完成 Z→PC。

（六）控制器设计

采用微程序设计的方法进行控制器设计。

6.1 微指令序列

每条机器指令对应的微指令序列在“节拍的划分”部分已给出。

6.2 微指令字长和微指令格式

如果将 CMDR 的下地址字段 Ad(CMDR)直接接到控制存储器的地址线上，并由下一个时钟周期的上升沿将该地址单元的内容（微指令）读到 CMDR 中，便能做到在一个时钟周期内读出并执行一条微指令。同理，也可将指令寄存器的操作码字段 OP(IR)经微地址形成部件形成的后续微地址的地址，直接送到控制存储器的地址线上。因此，可省去两个微操作 Ad(CMDR)→CMAR 和 OP(IR)→微地址形成部件→CMAR。

综上，10 条机器指令共对应 36 条微指令和 24 个微操作。根据直接编码方式，24 个微操作对应 24 位操作控制字段，36 条微指令对应 6 位下地址字段。微指令字长至少取 30 位。微指令格式如图 24 所示：

图 24 对应 10 条机器指令的微指令格式

其中，第 0 位表示控制　PC→MAR

第 1 位表示控制　1→R

第 2 位表示控制　M(MAR)→MDR

第 3 位表示控制　(PC+4)→PC

第 4 位表示控制　MDR→IR

第 5 位表示控制　R(IR[21~25])→Y

第 6 位表示控制　(Y)+(R(IR[16~20])→Z

第 7 位表示控制　(Y)-(R(IR[16~20])→Z

第 8 位表示控制　(Y)&(R(IR[16~20])→Z

第 9 位表示控制　(Y)|(R(IR[16~20])→Z

第 10 位表示控制　R(IR[16~20])→Y

第 11 位表示控制　(Y)<< (R(IR[6~10]))→Z

第 12 位表示控制　(Y)+ (R(IR[0~15]))→Z

第 13 位表示控制　(PC)+(R(IR[0~15]))<<2→Z

第 14 位表示控制　(Y)==(R(IR[16~20]))→ALU

第 15 位表示控制　(PC[28~31])||(R(IR[0~25]))||00→Z

第 16 位表示控制　Z→MAR

第 17 位表示控制　Z→PC

第 18 位表示控制　1→W

第 19 位表示控制　R(IR[16~20])→MDR

第 20 位表示控制　MDR→M(MAR)

第 21 位表示控制　Z→R(IR[11~15])

第 22 位表示控制　Z→R(IR[16~20])

第 23 位表示控制　MDR→R(IR[16~20])

6.3 微指令码点

微程序名称	微指令地址（八进制）	微指令（二进制代码）	
		操作控制字段	顺序控制字段

		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
取指	00	1	1																												1
	01			1	1																									1	
	02					1																				×	×	×	×	×	×
ADD	03						1																						1		
	04							1																					1		1
	05																						1								
SUB	06						1																						1	1	1
	07								1																			1			
	10																						1								
AND	11						1																					1			
	12									1																		1			1
	13																						1								
OR	14						1																					1		1	1
	15										1																	1	1		
	16																						1								
SLL	17											1															1				
	20												1														1				1
	21																						1								
ADDI	22						1																				1			1	1
	23													1													1		1		
	24																							1							
LW	25						1																				1		1	1	
	26													1													1		1	1	1
	27		1															1									1	1			
	30			1																							1	1			1
	31																								1						
BEQ	32						1								1												1	1		1	1
	33															1											1	1	1		
	34																		1												
SW	35						1																				1	1	1	1	
	36													1													1	1	1	1	1
	37																	1		1						1					
	40																				1					1					1

	41																			1										
J	42															1									1				1	1
	43																	1												

（七）总结

以上完成了指令格式设计，微操作定义、节拍划分、处理器设计、控制器设计五个部分。至此，已完成了基于 MIPS 指令系统的 RISC 处理器设计。