Table of Contents

1 Introduction

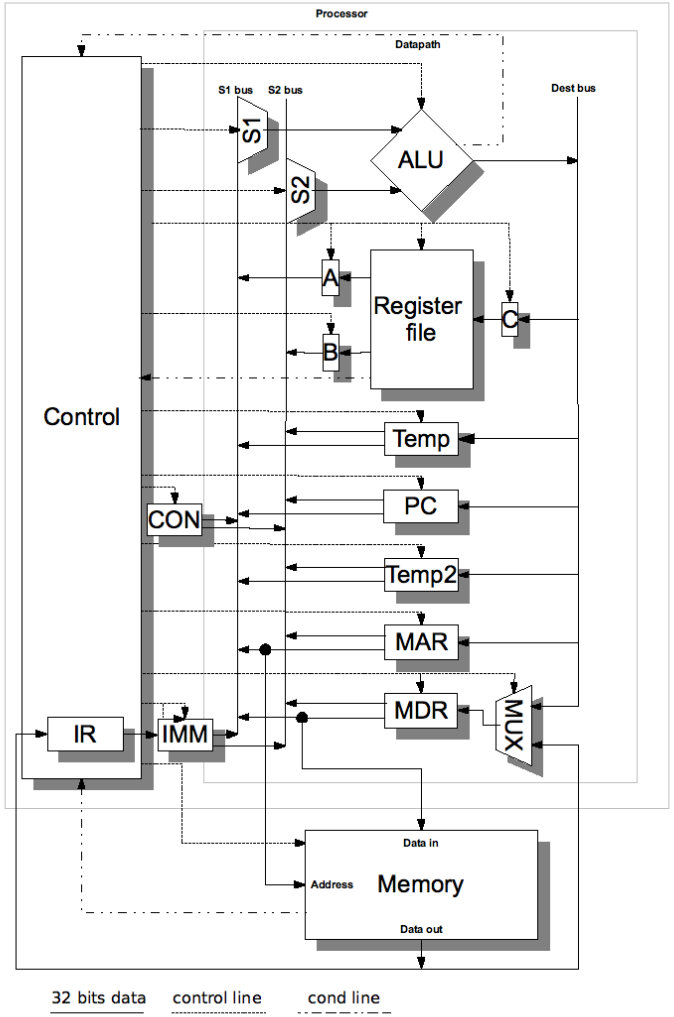
2.1 Design
2.2 Design choices

3 Components
- ALU
- Buffer
- Constant
- Immidiate parser
- Memory
- MUX
- Sx bus
- Register

4 Decode Tables
- Decode 1
- Decode 2
- Decode 3

5 Micro-code
5.1 Pseudo code
5.2 Binary code
5.3 Control program

6 Reference
6.1 Op- and func codes

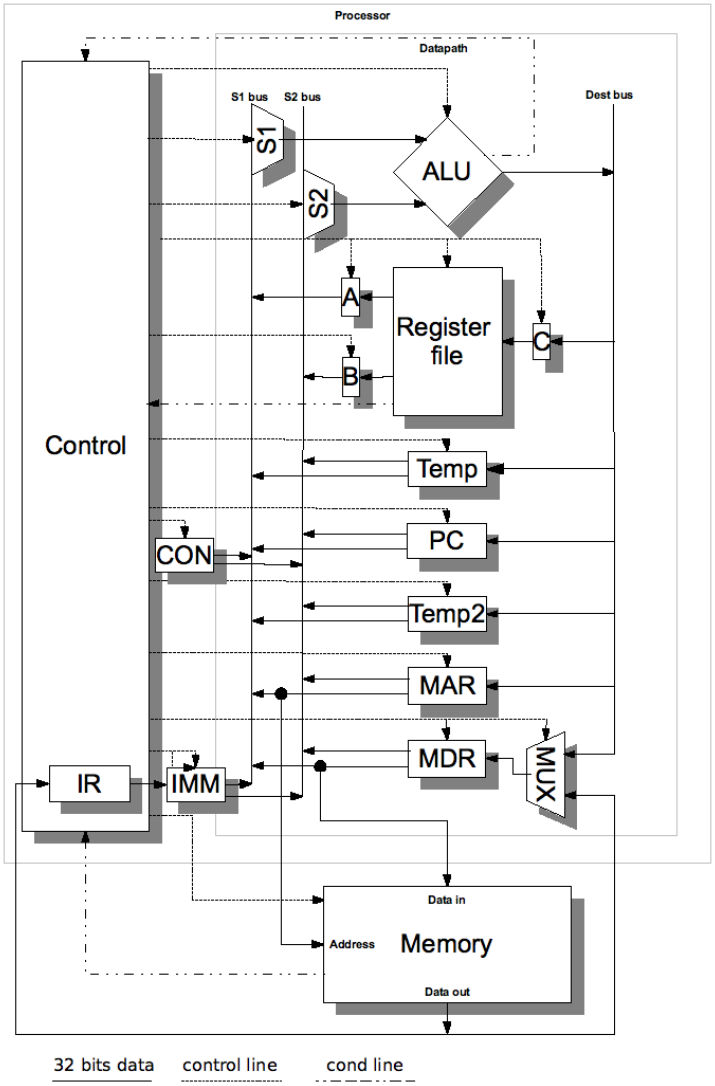


Creating a MIPS implementation from a DLX design specification takes a number of steps:
* Modify microcode to all MIPS instructions to passthrough, some more is to be told at DesignChoices
* Alter DLX design picture to include new components and signal, more details at Design
* Describe the components and their signals
* Describe the behaviour of the microcode program in pseudo code (Microcode)
* Describe the behaviour of the microcode program in byte code (MicrocodeBinary)
* Describe the behaviour of the control program in pseudo code (ControlCond)

Notation rules used:
* I[a ... b] = from input high-order bit number a till bit number b

Verbs used:

addr	=	address
ALU	=	Arithmetic Logical Unit
Bin	=	Binary
CON/Const	=	Constant Generator
Cond	=	Condition
cont/c	=	control
Dec	=	Decimal
Dest:	=	Destination
IMM	=	Immediate parser
IR	=	Instruction Registry
J-Label	=	Jump Label
MAR	=	Memory Address Registry
MDR	=	Memory Destination Registry
Misc	=	Miscellaneous
MPC	=	Control Program Counter
MUX	=	2-way multiplexer
PC	=	Program Counter
S1	=	n-way Multiplexer S1
S2	=	n-way Multiplexer S2
Uncond	=	Unconditional



In difference to the implementation of the DLX design there has been 3 components added namely:

* the constant generator, to generate constant values for example to raise the clock counter by 4 and more of this kind of operations

* the IMM, this small pieces of hardware is a hardwired version of the ALU operations SLL, SRA and SRL to make the value directly ready

* Temp2 has been added to make the instructions needed for both load and store operations, minimal

Both S1/S2/MUX has been modifed to actually looks like a multiplexer to better reflect their function

Signal lines have been added to give feedback of results and statuses

Choices made in my implementation of the MIPS instruction set:
* A '0' signal/bit means not active, not running. In other words negative set
* There will be no do not care signals in the implementation, excepts a the Decode tables. To make debugging crystal clear
* A SW will be preceded by a LW although this is not needed -and so will be a potential candidate for optimisation-
  it will make reading the code and flow more straight forward
* Documentation is mostly written in database style -lots of tables-, meaning checking could be done automatically
  , for example 80% of the code at MicrocodeBinary is filled dynamically to make sure inconsistencies and changes are done fast
* Signal lines always have the direction, from the component to the Control unit and their behaviour is described at the component
* Control lines always have the direction, from the control unit to the component and their behaviour is described at the component
* All data-lines are 32 bits lines
* Decode2 and Decode3 are written down on different sheets, but in reality both are combined to one big Decode table,
  cause they do not do not interfere each-other
* Overflow does not need to be taken into count, meaning some functions use exact the same code

Dec	Binary	Function	Meaning in C pseudo-code
0	0000	<none>	<none>
1	0001	ADD	C = A + B
2	0010	AND	C = A & B
3	0011	NOR	C = ~(A | B)
4	0100	OR	C = A | B
5	0101	Pass S1	C = A
6	0110	Pass S2	C = B
7	0111	SLL	C = A<<B
8	1000	SRA	
9	1001	SRL	C = A>>B
10	1010	SUB	C = A – B
11	1011	XOR	C = A ^ B
12	1100	<unused>	<unused>
13	1101	<unused>	<unused>
14	1110	<unused>	<unused>
15	1111	<unused>	<unused>

Description: Arithmetic Logical Unit, do computation of input X (and Y)
Signal in: <control> 4 bit

Signal Out:	<negative>	1 bit	Set if computed output is negative
	<zero>	1 bit	Set if Computed output is zero
Data out:	<databus>		
Data in:	S1, S2		

Dec	Bin	Notation	Meaning
0	0000	0	Do not accept new value
1	0001	1	Write value at Control in to buffer

Desc Small storage unit, able to store 32 bit
Signal in: <control> 1 bit
Control in: <variable input>
Control out: <variable output>

Known in design as: A, B, C, Temp, Temp2, PC, MAR,MDR

Dec	Bin	Notation	Dec Value	Binary Value			
0	0000	0	0	00000000	00000000	00000000	00000000
1	0001	1	1	00000000	00000000	00000000	00000001
2	0010	2	2	00000000	00000000	00000000	00000010
3	0011	3	3	00000000	00000000	00000000	00000011
3	0011	0x30,1x2	3	00000000	00000000	00000000	00000011
4	0100	4	4	00000000	00000000	00000000	00000100
5	0101	8	8	00000000	00000000	00000000	00001000
6	0110	16	16	00000000	00000000	00000000	00010000
7	0111	24	24	00000000	00000000	00000000	00011000
8	1000	1x4,0x28	8053063680	11110000	00000000	00000000	00000000
9	1001	1x8,0x24	8556380160	11111111	00000000	00000000	00000000
10	1010	1x16,0x16	8589803520	11111111	11111111	00000000	00000000
11	1011	<unused>	Err:502	<unused>	<unused>	<unused>	<unused>
12	1100	<unused>	Err:502	<unused>	<unused>	<unused>	<unused>
13	1101	<unused>	Err:502	<unused>	<unused>	<unused>	<unused>
14	1110	<unused>	Err:502	<unused>	<unused>	<unused>	<unused>
15	1111	<unused>	Err:502	<unused>	<unused>	<unused>	<unused>

Note: 3 has been put twice, to clearly identify the use cases of constant 3

Desc: Constant 32 bit value generator
Signal in: <control> 4 bits
Signal Out: <none>

Data out: <Sx-bus>
Data in: <none>

Dec	Binary	Function	Meaning	
0	00	<unused>	<unused>	
1	01	S16	Signed I[15...0]	SLL 16; SRA 16
2	10	SSA	Signed I[10...6]	SLL 21; SRA 27
3	11	U26	Unsigned I[25 ... 0]	SLL 6; SRL 6

Desc: IR code editor, to match specific part of the IR code and set it a the low-order side of the output
Signal in: <control> 2 bit
Signal Out: <none>

Data out: <S-bus>
Data in: IR

Dec	Binary	Function
0	00	<unused>
1	01	Read 32 bit from location MAR
2	10	Write 32 bit MDR to location MAR
3	11	<unused>

Desc: Memory, store up to '8 bit' bytes which could be specified by a 32 bit address
Signal in: <control> 2 bit
Signal Out: MemBusy 1 bit set if reading/writing data

Data out: IR/MUX
Data in: MDR and MAR

Note: The last 2 bits of the MAR are ignored (set '0') when reading out an address. The last 2 bits specify the offset so we do not need to take them into count

Dec	Bin	Notation	Meaning
0	0000	0	Choose value datapath
1	0001	1	Choose value Memory

Desc 2-way Multiplexer
Signal in: <control> 1 bit
Control in: Memory/<datapath>
Control out: MDR

Dec	Binary	Component
0	0000	<none>
1	0001	A
2	0010	B
3	0011	Temp
4	0100	Temp2
5	0101	PC
6	0110	MAR
7	0111	MDR
8	1000	Const
9	1001	IMM
10	1010	<none>
11	1011	<none>
12	1100	<none>
13	1101	<none>
14	1110	<none>
15	1111	<none>

Desc: 16-way multiplexer, do select the proper input for the ALU unit
Signal in: <control> 4 bit
Signal Out: <none>

Data out: ALU
Data in: <components>

Dec	Binary	Function
0	000	<none>
1	001	Read A
2	010	Read B
3	011	Read A and B
4	100	Write C
5	101	Read A, Write C
6	110	Read B, Write C
7	111	Read A and B, Write C

Description: Storage of 32 x 32 bits register entries
Signal in: <control> 3 bit
<addr A> 5 bit
<addr B> 5 bit
<addr C> 5 bit
Signal Out: <none>

Data out: A, B
Data in: C

Instruction (symbolically specified)	Absolute address	Label	Mask	Binary address
Memory	2	Mem:	1 X X X X X	0000010
Register function (op = 0x0)	14	Reg:	0 0 0 0 0 0	0001110
Immediate (op bit3 = 1)	15	Imm:	X X 1 X X X	0001111
Branch equal zero	5	BEQ:	0 0 0 1 0 0	0000101
Branch not equal zero	7	BNE:	0 0 0 1 0 1	0000111
Jump	11	J:	0 0 0 0 1 0	0001011
Jump And Link	9	JAL:	0 0 0 0 1 1	0001001

Decode table1 is used as first route on choice. It will match the mask defined on the O-code and output the corresponding binary address

instruction	Absolute address	Label	Binary address
ADDI	16	ADD/I/U:	0010000
ADDIU	16	ADD/I/U:	0010000
SLTI	33	SLTI/SLTIU:	0100001
SLTIU	33	SLTI/SLTIU:	0100001
ANDI	17	AND/I:	0010001
ORI	18	OR/I:	0010010
XORI	19	XOR/I:	0010011
LUI	28	LUI:	0011100
LB	55	LB:	0110111
LH	63	LH:	0111111
LW	71	LW:	1000111
LBU	59	LBU:	0111011
LHU	67	LHU:	1000011
SB	38	SB:	0100110
SH	45	SH:	0101101
SW	52	SW:	0110100

Decode table2 is used after the memory item has been loaded.
It will match the whole binary O-code and output the corresponding binary address

Instruction	Absolute address	Label	Binary address
SLL	22	SLL:	0010110
SRL	23	SRL:	0010111
SRA	24	SRA:	0011000
SLLV	25	SLLV:	0011001
SRLV	26	SRLV:	0011010
SRAV	27	SRAV:	0011011
JR	36	JR:	0100100
JALR	29	JALR:	0011101
BREAK	37	BREAK:	0100101
ADD	16	ADD/I/U:	0010000
ADDU	16	ADD/I/U:	0010000
SUB	21	SUB/U:	0010101
SUBU	21	SUB/U:	0010101
AND	17	AND/I:	0010001
OR	18	OR/I:	0010010
XOR	19	XOR/I:	0010011
NOR	20	NOR:	0010100
SLT	31	SLT/U:	0011111
SLTU	31	SLT/U:	0011111

Decode table3 is used after a R-Type code has been loaded (O-Code is 0x0) loaded.
It will match the whole binary R-code (I[5...0]) and output the corresponding binary address

Loc	Label	Dest	ALU	S1	S2	C	IMM	Misc	Cond	*J-Label*	Comment
0	Ifetch:	MAR	Pass S1	PC				Instr Read	MemBusy	Ifetch:	IR<-M[PC] wait for memory
1		PC	ADD	PC	Const	4		AB<-PF	Decode1		
2	Mem:	MAR	ADD	A	IMM		S16				Memory instruct.
3	Load:							Data read	MemBusy	Load:	Load MDR
4									Decode2/3		
5	BEQ:		SUB	A	B				ZeroValue	Branch:	Equal than branch
6									Uncond	Ifetch:	not = 0: not taken
7	BNE:		SUB	A	B				ZeroValue	Ifetch:	equal so fetch new entry
8	Branch:	PC	ADD	PC	IMM		S16		Uncond	Ifetch:	
9	JAL:	C	ADD	PC	Const	8					
10								R31 <-- C			Save PC + 8 into C
11	J:	PC	AND	PC	Const	1x4,0x28					zero out un-needed PC entries
12		Temp	SLL	IMM	Const	2	U26				Shifted left 2 bits
13		PC	OR	PC	Temp				Uncond	Ifetch:	Jump
14	Reg:	Temp	Pass S2		B				Decode2/3		source2 = reg
15	Imm:	Temp	Pass S2		IMM		S16		Decode2/3		source3 = imm.
16	ADD/I/U:	C	ADD	Temp	A				Uncond	Write:	combined with 4 entries
17	AND/I:	C	AND	A	Temp				Uncond	Write:	combined with 2 entries
18	OR/I:	C	OR	A	Temp				Uncond	Write:	combined with 2 entries
19	XOR/I:	C	XOR	A	Temp				Uncond	Write:	combined with 2 entries
20	NOR:	C	NOR	A	Temp				Uncond	Write:	
21	SUB/U:	C	SUB	A	Temp				Uncond	Write:	combined with 2 entries
22	SLL:	C	SLL	Temp	IMM		SSA		Uncond	Write:	
23	SRL:	C	SRL	Temp	IMM		SSA		Uncond	Write:	
24	SRA:	C	SRA	Temp	IMM		SSA		Uncond	Write:	
25	SLLV:	C	SLL	Temp	A				Uncond	Write:	
26	SRLV:	C	SRL	Temp	A				Uncond	Write:	
27	SRAV:	C	SRA	Temp	A				Uncond	Write:	
28	LUI:	C	SLL	Temp	Const	16			Uncond	Write:	Shift left 16 to zero extend
29	JALR:	C	ADD	PC	Const	8					
30		PC	Pass S1	A					Uncond	Write:	
31	SLT/U:		SUB	A	Temp				NegValue	Set1:	Set less than
32		C	Pass S2		Const	0			Uncond	Write:	
33	SLTI/SLTIU:		SUB	A	Temp				NegValue	Set1:	Set less than Unsigned
34		C	Pass S2		Const	0			Uncond	Write:	
35	Set1:	C	Pass S2		Const	1			Uncond	Write:	
36	JR:	PC	Pass S1	A					Uncond	Ifetch:	
37	BREAK:							Break signal	Uncond	BREAK:	
38	SB:	Temp	AND	MAR	Const	0x30,1x2					Byte position offset
39		Temp2	SLL	Temp	Const	3					Set the SRL value needed (* 8)
40		Temp	SLL	B	Const	24					get needed value
41		Temp	SRL	Temp	Temp2						set to proper location

42		Temp2	SRL	Const	Temp2	1x8,0x24				create whiper, and set to right location
43		MDR	XOR	MDR	Temp2					whipe needed places
44		MDR	OR	Temp	MDR			Uncond	Dloop:	combine both parts to the storage call
45	SH:	Temp	AND	MAR	Const	0x30,1x2				Byte position offset
46		Temp2	SLL	Temp	Const	3				Set the SRL value needed (* 8)
47		Temp	SLL	B	Const	16				get needed value
48		Temp	SRL	Temp	Temp2					set to proper location
49		Temp2	SRL	Const	Temp2	1x16,0x16				create whiper, and set to right location
50		MDR	XOR	MDR	Temp2					whipe needed places
51		MDR	OR	Temp	MDR			Uncond	Dloop:	store new value
52	SW:	MDR	Pass S1	B						Store needed value
53	Dloop:						Data write	MemBusy	Dloop:	Wait for mem to be ready
54								Uncond	Ifetch:	Fetch next instr
55	LB:	Temp	AND	MAR	Const	0x30,1x2				Byte position offset
56		Temp	SLL	Temp	Const	3				Set the SRL value needed (* 8)
57		Temp	SLL	MDR	Temp					Shift to the needed byte
58		C	SRA	Temp	Const	24		Uncond	Write:	Sign extend and write back
59	LBU:	Temp	AND	MAR	Const	0x30,1x2				Byte position offset
60		Temp	SLL	Temp	Const	3				Set the SRL value needed (* 8)
61		Temp	SLL	MDR	Temp					Shift to the needed byte
62		C	SRL	Temp	Const	24		Uncond	Write:	Shift and write back
63	LH:	Temp	AND	MAR	Const	0x30,1x2				Byte position offset
64		Temp	SLL	Temp	Const	3				Set the SRL value needed (* 8)
65		Temp	SLL	MDR	Temp					Shift to the needed byte
66		C	SRA	Temp	Const	16		Uncond	Write:	Sign extend and write back
67	LHU:	Temp	AND	MAR	Const	0x30,1x2				Byte position offset
68		Temp	SLL	Temp	Const	3				Set the SRL value needed (* 8)
69		Temp	SLL	MDR	Temp					Shift to the needed byte
70		C	SRL	Temp	Const	16		Uncond	Write:	Shift and write back
71	LW:	C	Pass S1	MDR						No Write, cause it is right below :-)
72	Write:							RtypeIR	WriteRd:	Choose which register to use
73							Rt <-- C	Uncond	Ifetch:	Write back Rt & go fetch next instruction
74	WriteRd:						Rd <-- C	Uncond	Ifetch:	Write back Rd & go fetch next instruction

	S1 Bus	S2 Bus	MUX	CONST	IMM	ALU	MEM	A	B	C	TEMP	TEMP2	PC	MAR	MDR	IR	Register				Cond	Jump Address
Type:	control	control	c	control	cont	control	cont	c	c	c	c	c	c	c	c	c	control	addr A	addr B	addr C	control	address
Bits:	3210	3210	0	3210	10	3210	10	0	0	0	0	0	0	0	0	0	210	43210	43210	43210	210	6543210
0	0101	0000	0	0000	00	0101	01	0	0	0	0	0	0	1	0	1	000	00000	00000	00000	010	0000000
1	0101	1000	0	0100	00	0001	00	1	1	0	0	0	1	0	0	0	011	I[25...21]	I[20...16]	00000	011	0000000
2	0001	1001	0	0000	01	0001	00	0	0	0	0	0	0	1	0	0	000	00000	00000	00000	000	0000000
3	0000	0000	1	0000	00	0000	01	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	010	0000011
4	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	100	0000000
5	0001	0010	0	0000	00	1010	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	101	0001000
6	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	001	0000000
7	0001	0010	0	0000	00	1010	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	101	0000000
8	0101	1001	0	0000	01	0001	00	0	0	0	0	0	1	0	0	0	000	00000	00000	00000	001	0000000
9	0101	1000	0	0101	00	0001	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	000	0000000
10	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	100	00000	00000	11111	000	0000000
11	0101	1000	0	1000	00	0010	00	0	0	0	0	0	1	0	0	0	000	00000	00000	00000	000	0000000
12	1001	1000	0	0010	11	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
13	0101	0011	0	0000	00	0100	00	0	0	0	0	0	1	0	0	0	000	00000	00000	00000	001	0000000
14	0000	0010	0	0000	00	0110	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	100	0000000
15	0000	1001	0	0000	01	0110	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	100	0000000
16	0011	0001	0	0000	00	0001	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
17	0001	0011	0	0000	00	0010	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
18	0001	0011	0	0000	00	0100	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
19	0001	0011	0	0000	00	1011	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
20	0001	0011	0	0000	00	0011	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
21	0001	0011	0	0000	00	1010	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
22	0011	1001	0	0000	10	0111	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
23	0011	1001	0	0000	10	1001	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
24	0011	1001	0	0000	10	1000	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
25	0011	0001	0	0000	00	0111	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
26	0011	0001	0	0000	00	1001	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000

27	0011	0001	0	0000	00	1000	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
28	0011	1000	0	0110	00	0111	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
29	0101	1000	0	0101	00	0001	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
30	0001	0000	0	0000	00	0101	00	0	0	0	0	0	1	0	0	000	00000	00000	00000	001	1001000
31	0001	0011	0	0000	00	1010	00	0	0	0	0	0	0	0	0	000	00000	00000	00000	110	0100011
32	0000	1000	0	0000	00	0110	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
33	0001	0011	0	0000	00	1010	00	0	0	0	0	0	0	0	0	000	00000	00000	00000	110	0100011
34	0000	1000	0	0000	00	0110	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
35	0000	1000	0	0001	00	0110	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
36	0001	0000	0	0000	00	0101	00	0	0	0	0	0	1	0	0	000	00000	00000	00000	001	0000000
37	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	000	00000	00000	00000	001	0100101
38	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	000	00000	00000	00000	000	0000000
39	0011	1000	0	0011	00	0111	00	0	0	0	0	1	0	0	0	000	00000	00000	00000	000	0000000
40	0010	1000	0	0111	00	0111	00	0	0	0	1	0	0	0	0	000	00000	00000	00000	000	0000000
41	0011	0100	0	0000	00	1001	00	0	0	0	1	0	0	0	0	000	00000	00000	00000	000	0000000
42	1000	0100	0	1001	00	1001	00	0	0	0	0	1	0	0	0	000	00000	00000	00000	000	0000000
43	0111	0100	0	0000	00	1011	00	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	000	0000000
44	0011	0111	0	0000	00	0100	00	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	001	0110101
45	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
46	0011	1000	0	0011	00	0111	00	0	0	0	0	1	0	0	0	0	000	00000	00000	00000	000	0000000
47	0010	1000	0	0110	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
48	0011	0100	0	0000	00	1001	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
49	1000	0100	0	1010	00	1001	00	0	0	0	0	1	0	0	0	0	000	00000	00000	00000	000	0000000
50	0111	0100	0	0000	00	1011	00	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	000	0000000
51	0011	0111	0	0000	00	0100	00	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	001	0110101
52	0010	0000	0	0000	00	0101	00	0	0	0	0	0	0	0	1	0	000	00000	00000	00000	000	0000000
53	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	010	0110101
54	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	001	0000000
55	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
56	0011	1000	0	0011	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
57	0111	0011	0	0000	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
58	0011	1000	0	0111	00	1000	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
59	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
60	0011	1000	0	0011	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
61	0111	0011	0	0000	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000

62	0011	1000	0	0111	00	1001	00	0	0	1	0	0	0	0	0	000	00000	00000	00000	001	1001000
63	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
64	0011	1000	0	0011	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
65	0111	0011	0	0000	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
66	0011	1000	0	0110	00	1000	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
67	0110	1000	0	0011	00	0010	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
68	0011	1000	0	0011	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
69	0111	0011	0	0000	00	0111	00	0	0	0	1	0	0	0	0	0	000	00000	00000	00000	000	0000000
70	0011	1000	0	0110	00	1001	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	001	1001000
71	0111	0000	0	0000	00	0101	00	0	0	1	0	0	0	0	0	0	000	00000	00000	00000	000	0000000
72	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	000	00000	00000	00000	111	1001010
73	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	100	00000	00000	I[20...16]	001	0000000
74	0000	0000	0	0000	00	0000	00	0	0	0	0	0	0	0	0	0	100	00000	00000	I[15...11]	001	0000000

Dec	Binary	Function	Meaning
0	000	Next	MPC = MPC + 1
1	001	Uncond	MPC = J-Label
2	010	MemBusy	MPC = J-Label if MemBusy else Next
3	011	Decode1	MPC is calculated from Decode1
4	100	Decode2/3	MPC is calculated from Decode2&Decode3
5	101	ZeroValue	MPC = J-Label if ZeroValue else Next
6	110	NegValue	MPC = J-Label if NegValue else Next
7	111	RtypeIR	MPC = J-Label if R-Type (meaning I[31...26] = 00000) else Next

The behaviour of the control program is pretty straight forward:
* At the start we start running at instruction 0x0 (MPC = 0x0)
* We 'issue' the command by sending all control signals to the different components
* We check the cond and calculate the new value corresponding the table below

Instruction	O-code (hex)	Type	O-code	5 bits	5 bits	5 bits	5bits	6 bits	Textual
			I[31...26]	I[25...21]	I[20...16]	I[15...11]	I[10...6]	I[5...0]	
			5 4 3 2 1 0					5 4 3 2 1 0	
J	0x2	J	0 0 0 0 1 0	target1	target2	target3	target4	target5	Jump
JAL	0x3	J	0 0 0 0 1 1	target1	target2	target3	target4	target5	Jump And Link
BEQ	0x4	I	0 0 0 1 0 0	rs	rt	offset1	offset2	offset3	Branch on EQual
BNE	0x5	I	0 0 0 1 0 1	rs	rt	offset1	offset2	offset3	Branch on Not Equal
ADDI	0x8	I	0 0 1 0 0 0	rs	rt	imm1	imm2	imm3	Add Immediate
ADDIU	0x9	I	0 0 1 0 0 1	rs	rt	imm1	imm2	imm3	Add Immediate Unsigned
SLTI	0x0a	I	0 0 1 0 1 0	rs	rt	imm1	imm2	imm3	Set on Less Than Immediate
SLTIU	0x0b	I	0 0 1 0 1 1	rs	rt	imm1	imm2	imm3	Set on Less Than Immediate Unsigned
ANDI	0x0c	I	0 0 1 1 0 0	rs	rt	imm1	imm2	imm3	AND Immediate
ORI	0x0d	I	0 0 1 1 0 1	rs	rt	imm1	imm2	imm3	OR Immediate
XORI	0x0e	I	0 0 1 1 1 0	rs	rt	imm1	imm2	imm3	eXclusive OR Immediate
LUI	0x0f	I	0 0 1 1 1 1	rs	rt	imm1	imm2	imm3	Load Upper Immediate
LB	0x20	I	1 0 0 0 0 0	base	rt	offset1	offset2	offset3	Load Byte
LH	0x21	I	1 0 0 0 0 1	base	rt	offset1	offset2	offset3	Load Halfword
LW	0x23	I	1 0 0 0 1 1	base	rt	offset1	offset2	offset3	Load Word
LBU	0x24	I	1 0 0 1 0 0	base	rt	offset1	offset2	offset3	Load Byte Unsigned
LHU	0x25	I	1 0 0 1 0 1	base	rt	offset1	offset2	offset3	Load Halfword Unsigned
SB	0x28	I	1 0 1 0 0 0	base	rt	offset1	offset2	offset3	Store Byte
SH	0x29	I	1 0 1 0 0 1	base	rt	offset1	offset2	offset3	Store Halfword
SW	0x2b	I	1 0 1 0 1 1	base	rt	offset1	offset2	offset3	Store Word
SLL	0x0	R	0 0 0 0 0 0	rs	rt	rd	sa	0 0 0 0 0 0	Shift Left Logical
SRL	0x2	R	0 0 0 0 0 0	rs	rt	rd	sa	0 0 0 0 1 0	Shift Right Logical
SRA	0x3	R	0 0 0 0 0 0	0	rt	rd	sa	0 0 0 0 1 1	Shift Right Arithmetic
SLLV	0x4	R	0 0 0 0 0 0	rs	rt	rd	0	0 0 0 1 0 0	Shift Left Logical Variable
SRLV	0x6	R	0 0 0 0 0 0	rs	rt	rd	0	0 0 0 1 1 0	Shift Right Logical Variable
SRAV	0x7	R	0 0 0 0 0 0	rs	rt	rd	0	0 0 0 1 1 1	Shift Right Arithmetic Variable
JR	0x8	R	0 0 0 0 0 0	rs	0	0	0	0 0 1 0 0 0	Jump Register
JALR	0x9	R	0 0 0 0 0 0	rs	0	rd	0	0 0 1 0 0 1	Jump And Link Register
BREAK	0x0d	R	0 0 0 0 0 0	code1	code2	code3	code4	0 0 1 1 0 1	Breakpoint

ADD	0x20	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 0 0 0	ADD
ADDU	0x21	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 0 0 1	ADD Unsigned
SUB	0x22	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 0 1 0	SUBtract
SUBU	0x23	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 0 1 1	SUBtract Unsigned
AND	0x24	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 1 0 0	AND
OR	0x25	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 1 0 1	OR
XOR	0x26	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 1 1 0	eXclusive OR
NOR	0x27	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 0 1 1 1	NOR
SLT	0x2a	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 1 0 1 0	Set on Less Than
SLTU	0x2b	R	0 0 0 0 0 0	rs	rt	rd	0	1 0 1 0 1 1	Set on Less Then Unsigned

Dest
Source1
Source2
Ignored