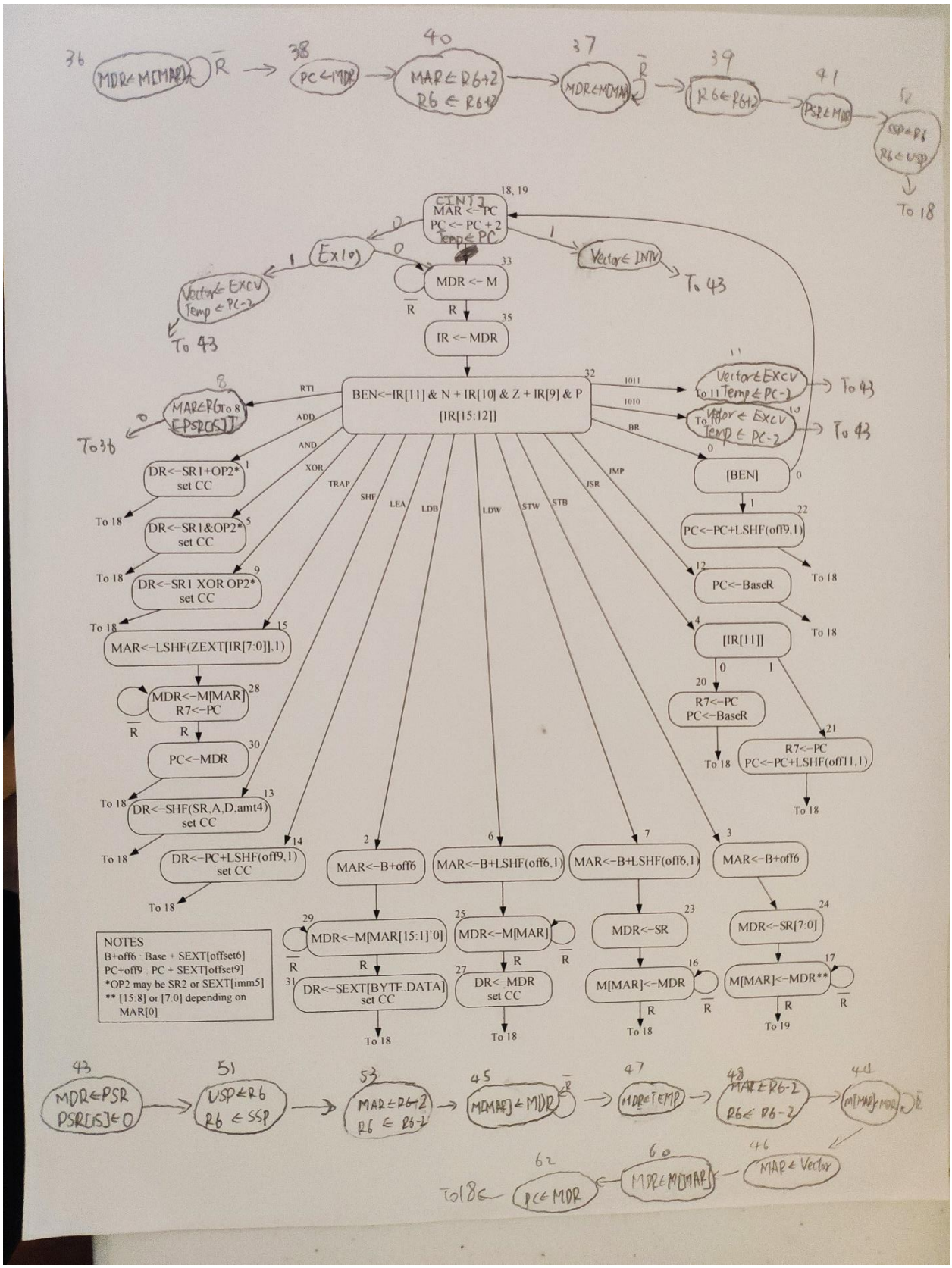
36
MDR←M[MAR]
R
38
PC←MDR
40
MAR←R6+2
R6←R6+2
37
MDR←M[MAR]
R
39
R6←R6+2
41
PSR←MDR
42
SSP←R6
R6←USP
To 18
INT
18, 19
MAR <- PC
PC <- PC + 2
Temp ← PC
0
1
Ex(0)
Vector← INTV
To 43
Vector← EXCV
Temp ← PC-2
To 43
0
MDR <- M
R
R
35
IR <- MDR
8
MAR←R6 To 8
PSR[15]
To 36
RTI
32
BEN<-IR[11] & N + IR[10] & Z + IR[9] & P
[IR[15:12]]
1011
1010
BR
11
Vector←EXCV
To 11 Temp ← PC-1
To 43
10
Vector ← EXCV
Temp ← PC-2
To 43
0
[BEN]
0
1
22
PC←PC+LSHF(off9,1)
To 18
12
PC←BaseR
To 18
4
[IR[11]]
0
1
20
R7←PC
PC←BaseR
To 18
21
R7←PC
PC←PC+LSHF(off11,1)
To 18
ADD
AND
XOR
TRAP
SHF
LEA
LDB
LDW
STW
STB
JMP
JSR
1
DR<-SR1+OP2*
set CC
To 18
5
DR<-SR1&OP2*
set CC
To 18
9
DR<-SR1 XOR OP2*
set CC
To 18
15
MAR<-LSHF(ZEXT[IR[7:0]],1)
28
MDR<-M[MAR]
R7<-PC
R
R
30
PC<-MDR
To 18
13
DR<-SHF(SR,A,D,amt4)
set CC
To 18
14
DR<-PC+LSHF(off9,1)
set CC
To 18
2
MAR<-B+off6
6
MAR<-B+LSHF(off6,1)
7
MAR<-B+LSHF(off6,1)
3
MAR<-B+off6
29
MDR<-M[MAR[15:1]'0]
R
R
31
DR<-SEXT[BYTE.DATA]
set CC
To 18
25
MDR<-M[MAR]
R
R
27
DR<-MDR
set CC
To 18
23
MDR<-SR
16
M[MAR]<-MDR
R
R
To 18
24
MDR<-SR[7:0]
17
M[MAR]<-MDR**
R
R
To 19
NOTES
B+off6 : Base + SEXT[offset6]
PC+off9 : PC + SEXT[offset9]
*OP2 may be SR2 or SEXT[imm5]
** [15:8] or [7:0] depending on
MAR[0]
43
MDR←PSR
PSR[15]←0
51
USP←R6
R6 ← SSP
53
MAR←R6-2
R6 ← R6-1
45
M[MAR]←MDR
R
47
MDR←TEMP
48
MAR←R6-2
R6←R6-2
49
M[MAR]←MDR
R
46
MAR←Vector
60
MDR←M[MAR]
62
PC←MDR
To 18

The interrupt and exception are implemented by states 43-62, which is shown at the bottom of the page. In these states, user stack pointer is saved. PC and PSR are pushed onto the supervisor stack and the supervisor mode is changed. Then the appropriate vector is loaded into MAR and PC is loaded with the starting address of the handler. States 36-52 on the top implement the RTI instruction. They pop PC and PSR from the supervisor stack and switch R6 back to the user stack pointer. States 18 and 19 now also store the old PC in a temp register and branch based on INT, which is set for an interrupt.

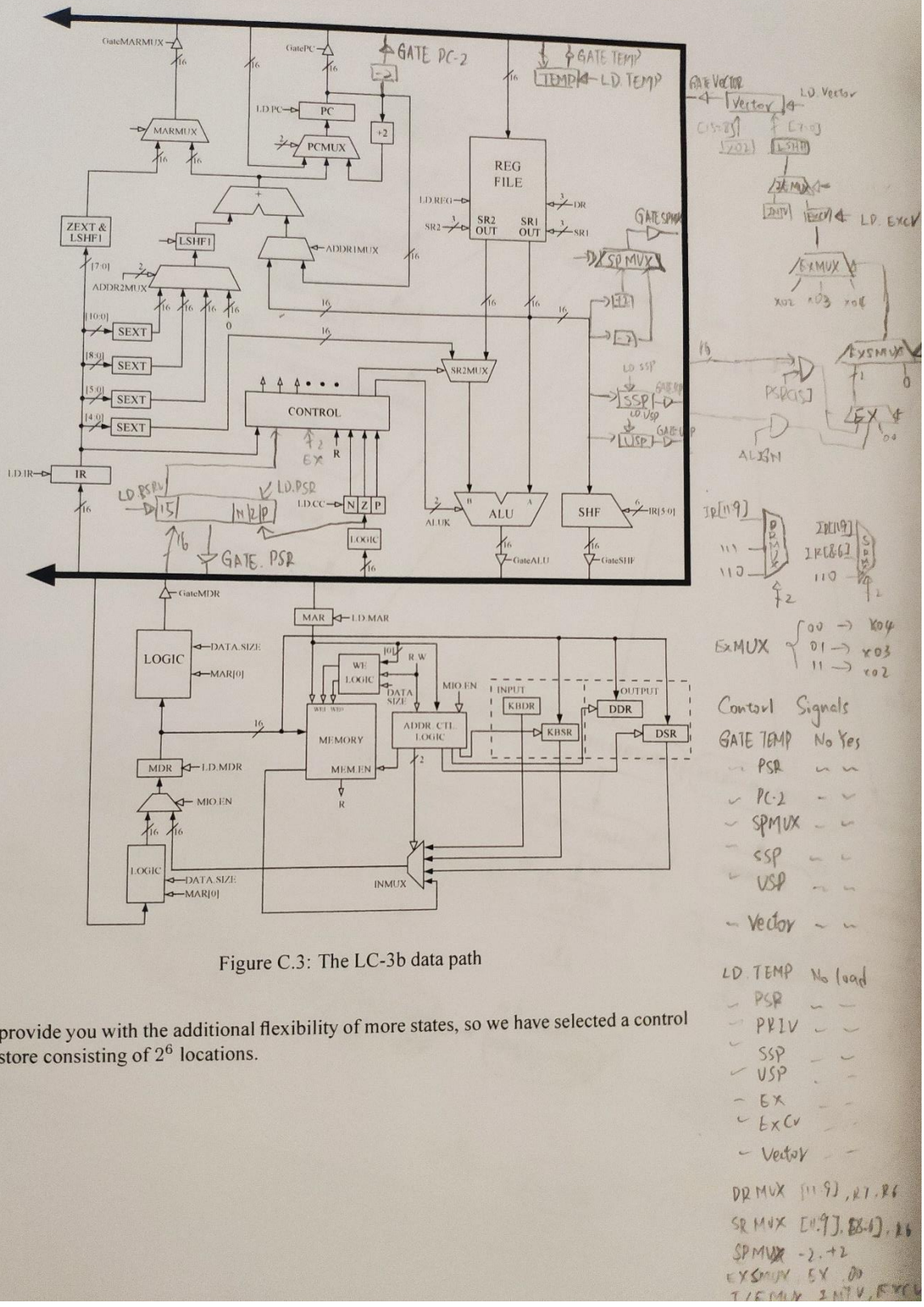

Figure C.3: The LC-3b data path

provide you with the additional flexibility of more states, so we have selected a control store consisting of 2^6 locations.

SSP is a register which is used to store the supervisor stack pointer while USP is for storing user stack pointer. SPMUX is used to take an input from SR1 of the register file and select between SR1+2 and SR1-2. It can also drive the bus. TEMP is a register which is used to store PC before it is pushed onto the stack when going to a handler. The PSR register is added to store the PSR. The PSR [15] bit can be loaded with 0 directly with LD.PRIV. PSR[2:0] store the condition codes by connecting to the input of the condition code register. The other added structure (on the right of the picture) is used to detect exceptions and load the correct vector for interrupts and exceptions. EX is a 2-bit register which is set by an exception. EXSMUX selects 0 or EX based on the control signal. The output of EXSMUX is connected to EXMUX, which is used to select the vector based on which exception is detected, and the result is stored in EXCV if LD.EXCV is set. I/EMUX selects between the interrupt and exception vector to be loaded.

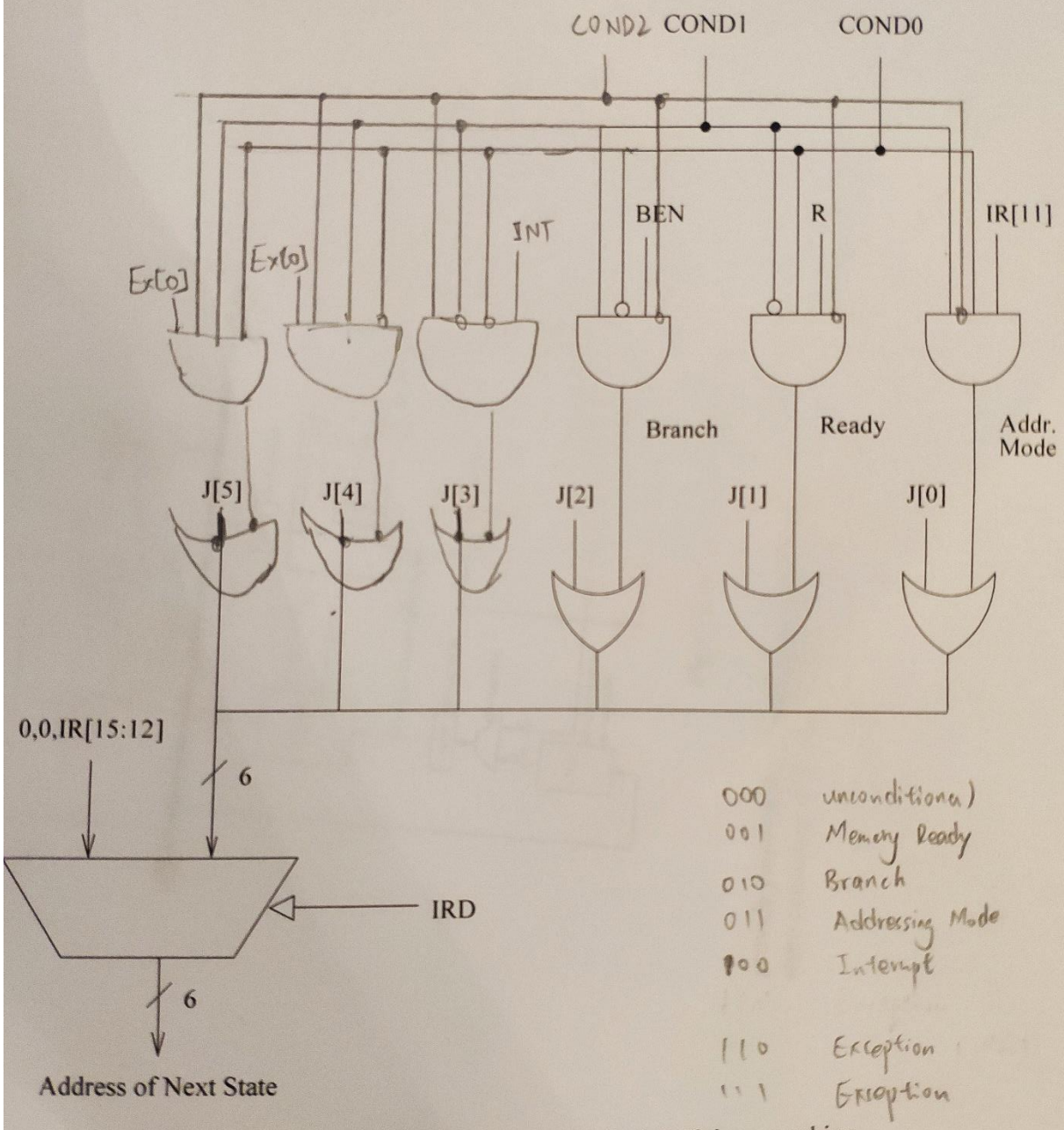

Figure C.5: The microsequencer of the LC-3b base machine

unused opcodes, the microarchitecture would execute a sequence of microinstructions, starting at state 10 or state 11, depending on which illegal opcode was being decoded. In both cases, the sequence of microinstructions would respond to the fact that an instruction with an illegal opcode had been fetched.

In this design, we have more conditions than before. Only 2 condition bits are not enough so we have to add one more condition bit. The six conditions are shown in the picture.