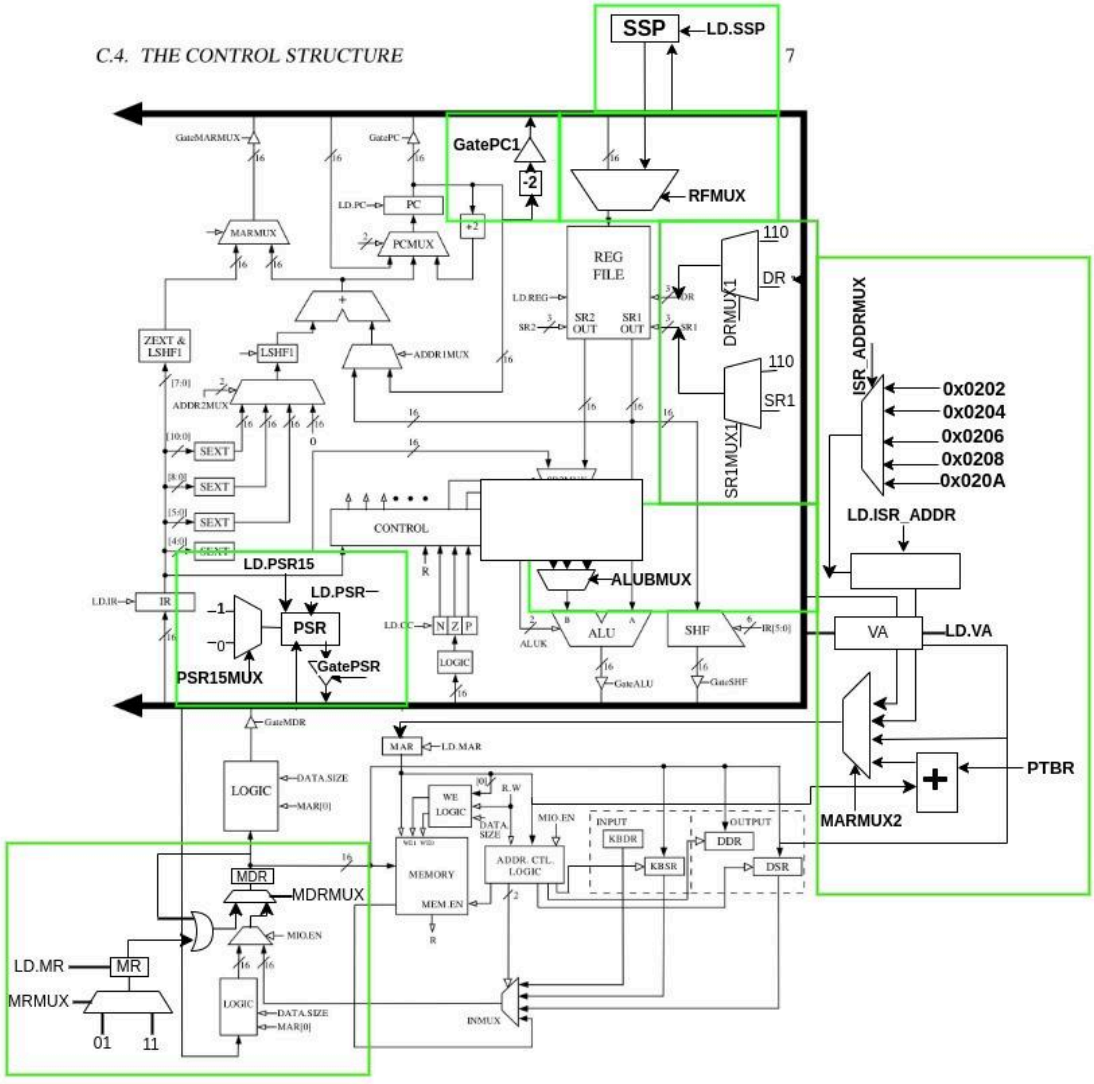
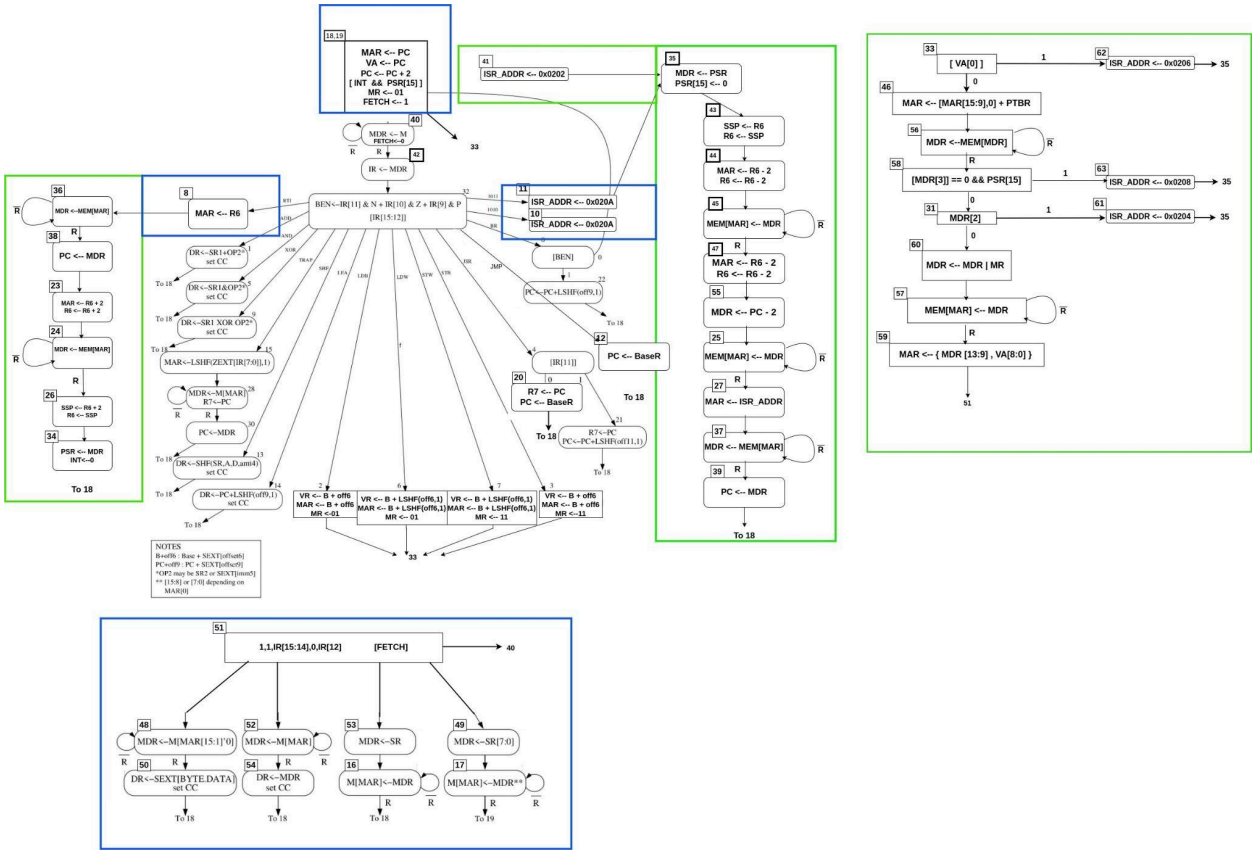


Figure C.3: The LC-3b data path

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

An SSP Register, controlled by the LD.SSP signal is added. This register captures data from the bus and serves as an input to the RFMUX, which toggles between bus data and the SSP Register output to load SSP data into R6.
GatePC1 transmits the Program Counter (PC), decremented by 2, onto the bus.
The ALUBMUX selects between src2, +2, and -2, facilitating the increment or decrement of R6 by 2.
SR1MUX1 and DRMUX1 set the source and destination register to R6, respectively, enabling R6 to function as a stack pointer.
ISR_ADDRMUX switches between 0x0202, 0x0204, 0x0206, 0x0208, and 0x020A (Address in the interrupt vector table), depending on the exception or interrupt, loading the selected address into ISR_ADDR when LD.ISR_ADDR is high.
MARMUX2 selects between bus data, ISR_ADDR, Physical Address, and Address of the PTE output to feed into the MAR register, aiding in ISR address loading.
PSR15MUX manages the PSR[15] bit, indicating user vs. supervisor privilege. LD.PSR15 loads PSR15MUX's output into PSR[15]. Additionally, LD.PSR loads the PSR from the bus, while GatePSR outputs the PSR onto the bus.
MDRMUX selects between MDR and modified MDR
MR register keeps track of whether it is a load or store operation



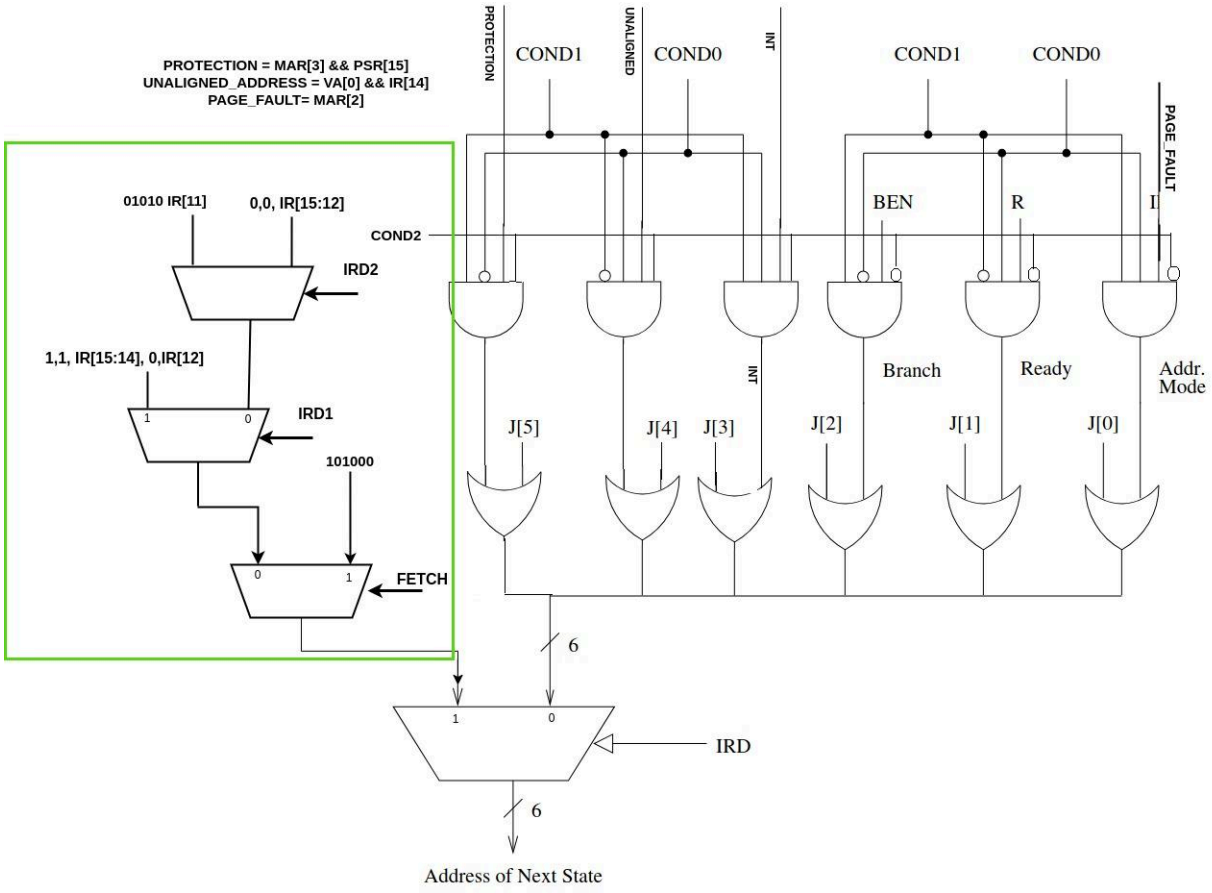
States 8, 36, 38, 23, 24, 26, and 34 implement the RTI (Return from Interrupt) instruction.

- **States 8, 36, and 38**: Pop the program counter (PC) stored at the stack pointer (R6).
- **State 23**: Increment the stack pointer by 2.
- **State 26:** Restores the R6 value
- **States 24 and 34**: Pop the program status register (PSR) and disable interrupts.

States 33, 58, and 31 check for unaligned address, protection, and page fault exceptions, respectively. All of them diverge to starting **State 35**, which is responsible for storing the PC and PSR on the stack in case of an interrupt or exception, via **States 62, 63 and 61**, respectively.

- **State 41**: Contains the address of the timer interrupt service routine (ISR) in the interrupt vector table.
- **State 63**: Contains the address of the protection exception ISR in the interrupt vector table.
- **State 62**: Contains the address of the unaligned address exception ISR in the interrupt vector table.
- **State 61**: Contains the address of the page fault exception ISR in the interrupt vector table.
- **State 11**: Contains the address of the unknown opcode exception in the interrupt vector table.

State 51 is executed if there are no interrupts or exceptions. It transitions to **State 48 (if LDB), State 52 (if LDW), State 53 (if STW), State 49 (if STB) and State 40 (if Fetch)**



Added IRD1 Mux to select states i.e. 48, 52, 53, and 49 from the state 31. The IR[13] (2nd bit) is tied to 0 to enable state transitions 48 → 50 and 52 → 54.

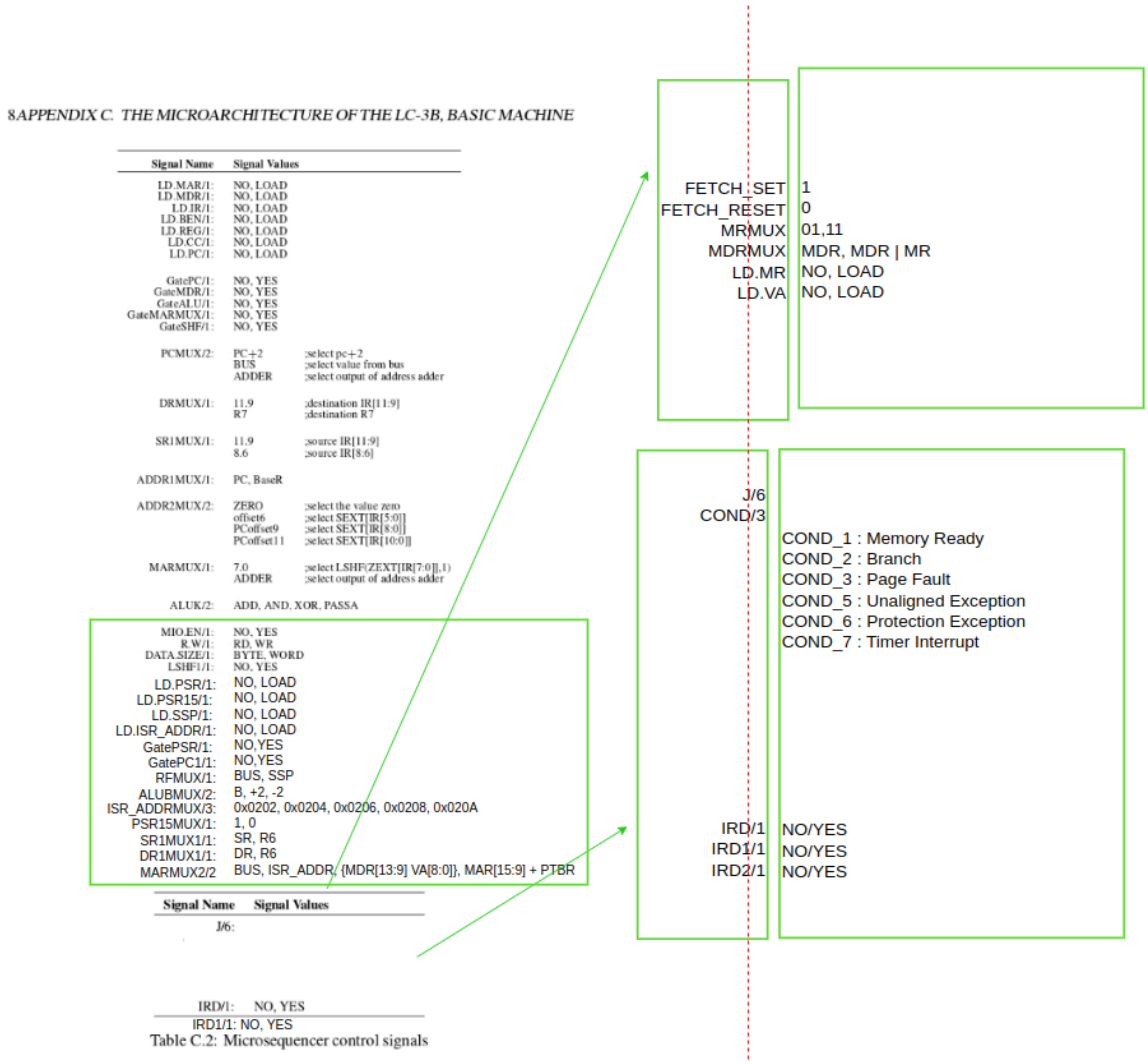
Added IRD2 Mux is added to enable the state transition for jump instruction.

The Fetch Mux has been added to facilitate the transition to State 40 during the instruction fetch process.

Requirement → Protection, Unaligned Address, and Interrupt Conditions
The above requirements led to the addition of the COND2 signal.

COND = 110 (Protection Exception)
COND = 101 (Unaligned Address Exception)
COND = 111 (Timer Interrupt)

The Unaligned Address Exception is triggered when VA[0] == 1
The Protection Exception is triggered if the MDR[3] == 0 i.e. the page is protected
The Protection Exception is triggered if the MDR[2] == 0 i.e. the page isn't valid
The Timer Interrupt is triggered when the interrupt vector is 0x01 and the system is in user mode (i.e., PSR[15] = 1).



8 APPENDIX C. THE MICROARCHITECTURE OF THE LC-3B, BASIC MACHINE

Signal Name	Signal Values	
LD.MAR/1:	NO, LOAD	
LD.MDR/1:	NO, LOAD	
LD.IR/1:	NO, LOAD	
LD.BEN/1:	NO, LOAD	
LD.REG/1:	NO, LOAD	
LD.CC/1:	NO, LOAD	
LD.PC/1:	NO, LOAD	
GatePC/1:	NO, YES	
GateMDR/1:	NO, YES	
GateALU/1:	NO, YES	
GateMARMUX/1:	NO, YES	
GateSHF/1:	NO, YES	
PCMUX/2:	PC+2	;select pc+2
	BUS	;select value from bus
	ADDER	;select output of address adder
DRMUX/1:	11.9	;destination IR[11:9]
	R7	;destination R7
SR1MUX/1:	11.9	;source IR[11:9]
	8.6	;source IR[8:6]
ADDR1MUX/1:	PC, BaseR	
ADDR2MUX/2:	ZERO	;select the value zero
	offset6	;select SEXT[IR[5:0]]
	PCoffset9	;select SEXT[IR[8:0]]
	PCoffset11	;select SEXT[IR[10:0]]
MARMUX/1:	7.0	;select LSHF(ZEXT[IR[7:0]],1)
	ADDER	;select output of address adder
ALUK/2:	ADD, AND, XOR, PASSA	
MIO.EN/1:	NO, YES	
R.W/1:	RD, WR	
DATA.SIZE/1:	BYTE, WORD	
LSHF1/1:	NO, YES	
LD.PSR/1:	NO, LOAD	
LD.PSR15/1:	NO, LOAD	
LD.SSP/1:	NO, LOAD	
LD.ISR_ADDR/1:	NO, LOAD	
GatePSR/1:	NO,YES	
GatePC1/1:	NO,YES	
RFMUX/1:	BUS, SSP	
ALUBMUX/2:	B, +2, -2	
ISR_ADDRMUX/3:	0x0202, 0x0204, 0x0206, 0x0208, 0x020A	
PSR15MUX/1:	1, 0	
SR1MUX1/1:	SR, R6	
DR1MUX1/1:	DR, R6	
MARMUX2/2	BUS, ISR_ADDR, {MDR[13:9] VA[8:0]}, MAR[15:9] + PTBR	
J/6:		
IRD/1:	NO, YES	
IRD1/1:	NO, YES	

Table C.2: Microsequencer control signals

FETCH_SET 1
FETCH_RESET 0
MRMUX 01,11
MDRMUX MDR, MDR | MR
LD.MR NO, LOAD
LD.VA NO, LOAD

J/6
COND/3

COND_1 : Memory Ready
COND_2 : Branch
COND_3 : Page Fault
COND_5 : Unaligned Exception
COND_6 : Protection Exception
COND_7 : Timer Interrupt

IRD/1 NO/YES
IRD1/1 NO/YES
IRD2/1 NO/YES

LD.PSR → Loads the bus data on the PSR
LD.PSR15 → Loads either 0 or 1 on the PSR[15] bit

LD.SSP → Loads the bus data on the SSP register.
LD.ISR_ADDR → Loads the ISR_ADDRMUX output on the ISR_ADDR register
GatePSR → Loads the PSR register data on the bus.
GatePC → Loads the PC register data on the bus
RFMUX → Select between SSP and bus as Register File Input
ALUBMUX → Select between Reg File 2nd output, +2, and -2 as an ALU input
ISR_ADDRMUX →Select between 0x0202, 0x204, 0x0206, 0x0208, and 0x020A as ISR address
PSR15MUX →Select between 0 and 1 for the PSR[15] bit
SR1MUX1 → Select between SRMUX output and R6 as a source register
DR1MUX1 → Selects between DRMUX output and R6 as the destination register
MARMUX2 → Selects between data on the bus, ISR_ADDR register, Physical Address, Address of PTE

FETCH_SET → set the fetch register
FETCH_RESET → Reset the fetch register
MRMUX →Select between 01 and 11
MDRMUX → Select between MDR and MDR | MR (Modified MDR)
LD.MR → Load MR Register
LD.VA → Load VA Register