2 Microcoded PARCv1 Processor

2.a Implementing Conditional Move

		Bus Enables					Register Enables						Mux		Func		RF		mreq		
State	**Pseudo Control Sigs**	**pc**	**iau**	**alu**	**rf**	**rd**	**pc**	**ir**	**a**	**b**	**c**	**wd**	**b**	**c**	**iau**	**alu**	**sel**	**wen**	**val**	**op**	**next**
M0:	A <- RF[rt]	0	0	0	1	0	0	0	1	0	0	0	x	x	x	x	rt	0	0	x	n
M1:	B <- RF[r0]	0	0	0	1	0	0	0	0	1	0	0	b	x	x	x	r0	0	0	x	n
M2:	A<-RF[rs];goto F0 if A=B	0	0	1	1	0	0	0	1	0	0	0	x	x	x	cmp	rs	0	0	x	b
M3:	RF[rd] <- A	0	0	1	1	0	0	0	0	0	0	0	x	x	x	cpa	rd	1	0	x	f

Figure 2: Microcode table for movn

M0 and M1 puts the values of rt and r0 into registers A and B respectively for comparison by the ALU. This is to check and see if rt is 0. M2 uses the cmp function of the ALU to check if rt is 0 and goto F0 (skip to next instruction) if rt is 0 by setting the next state to b. If rt is not 0, then the FSM will continue to execute states M3 and M4. M3 gets the value of rs and temporarily puts it in register A. M4 moves the value in register A to rd by utilizing the cpa (copy A to output) function of the ALU. This completes the microcode program for movn.

2.b Implementing Memory-Memory Increment Instruction

		Bus Enables					Register Enables						Mux		Func		RF		mreq		
State	**Pseudo Control Sigs**	**pc**	**iau**	**alu**	**rf**	**rd**	**pc**	**ir**	**a**	**b**	**c**	**wd**	**b**	**c**	**iau**	**alu**	**sel**	**wen**	**val**	**op**	**next**
I0:	C <- si(imm)	0	0	0	0	0	0	1	0	0	1	0	x	b	si	x	x	0	0	x	n
I1:	B <- RF[rs]	0	0	0	1	0	0	0	0	1	0	0	b	x	x	x	rs	0	0	x	n
I2:	A <- RF[r0]	0	0	0	1	0	0	0	1	0	0	0	x	x	x	x	r0	0	0	x	n
I3:	C[0]? A+B:copy A	0	0	1	0	0	0	0	1	1	1	0	s	s	x	+?	x	0	0	x	n
I4:	C[0]? A+B:copy A	0	0	1	0	0	0	0	1	1	1	0	s	s	x	+?	x	0	0	x	n
I5:	C[0]? A+B:copy A	0	0	1	0	0	0	0	1	1	1	0	s	s	x	+?	x	0	0	x	n
I6:	C[0]? A+B:copy A	0	0	1	0	0	0	0	1	0	0	0	x	x	x	+?	x	0	0	x	n
I7:	B <- RF[rt]	0	0	0	1	0	0	0	0	1	0	0	b	x	x	x	rt	0	0	x	n
I8:	mreq_addr<-B<-A+B	0	0	1	0	0	0	0	0	1	0	0	b	x	x	+	x	0	1	r	n
I9:	A <- RD	0	0	0	0	1	0	0	1	0	0	0	x	x	x	x	x	0	0	x	n
I10:	WD <- A+1	0	0	1	0	0	0	0	0	0	0	1	x	x	x	+1	x	0	0	x	n
I11:	mreq_addr <- B	0	0	1	0	0	0	0	0	0	0	0	x	x	x	cpb	x	0	1	w	f

Figure 3: Microcode table for inc

The microinstruction I0 to I2 sets up registers A, B, and C for doing the R[rs] x imm[3:0] multiplication. I3 to I7 uses the +? function of the ALU to perform the multiplication. I7 gets the rt register to prepare for the addition. I8 performs the addition to calculate the memory address. It send that address to both mreq_addr and register B for storage. I9 gets the value returned from the memory and puts it into register A. I10 performs the increment by 1 function and places the result into the WD register. I11 sends the memory address stored in B to the mreq_req address so that the result in WD can be stored back into memory.