ECEN 240: Final Project Instruction Set and Decoder

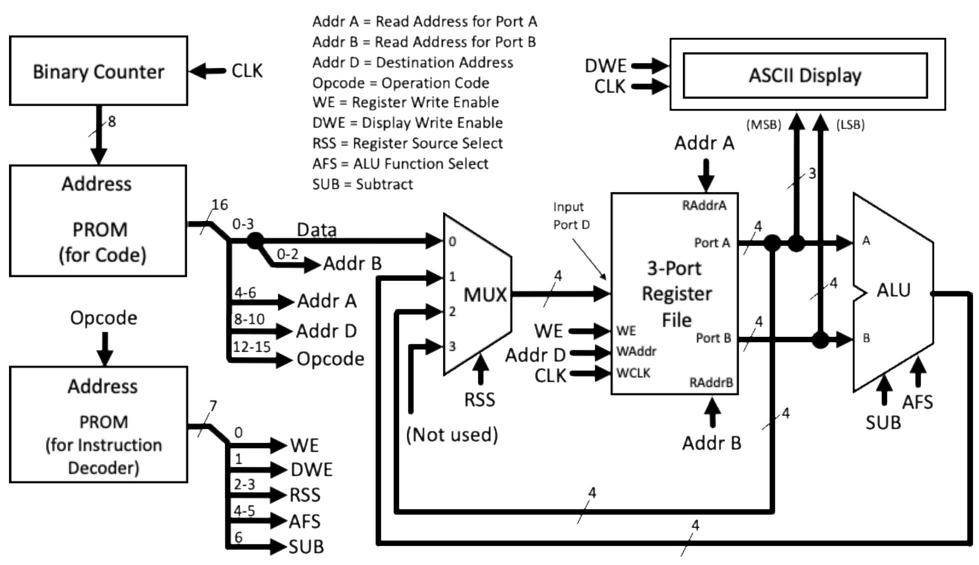

Figure 1 – A 4-bit CPU Block Diagram

Microprocessor Instruction Set

Large microprocessors have dozens (or even hundreds) of instructions (operations) in their instruction set. The number of bits required for each instruction will vary from processor to processor. This microprocessor architecture has nine instructions which may be represented in assembly code format or in 16-bit machine code format (see Table 1).

Opcode	Mnemonic	Instruction Operation	Assembly Code Format	Machine Code Format (16-bits)
0000	NOP	No Operation	NOP	0000 0000 0000 0000
0001	LD	Rd <= Data	LD Rd, Data	0001 0ddd 0000 DDDD
0010	MOV	Rd <= Ra	MOV Rd, Ra	0010 0ddd 0aaa 0000
0011	DISP	Display {Ra, Rb} on ASCII display	DISP Ra, Rb	0011 0000 0aaa 0bbb
0100	XOR	Rd <= Ra XOR Rb	XOR Rd, Ra, Rb	0100 0ddd 0aaa 0bbb
0101	AND	Rd <= Ra AND Rb	AND Rd, Ra, Rb	0101 0ddd 0aaa 0bbb
0110	OR	Rd <= Ra OR Rb	OR Rd, Ra, Rb	0110 0ddd 0aaa 0bbb
0111	ADD	Rd <= Ra + Rb	ADD Rd, Ra, Rb	0111 0ddd 0aaa 0bbb
1111	SUB	Rd <= Ra - Rb	SUB Rd, Ra, Rb	1111 0ddd 0aaa 0bbb

		Ra = Contents of register specified by Addr A
		Rb = Contents of register specified by Addr B
		Rd = Register specified by Addr D
		Data = Direct data from the program PROM
				a = Port A Address bit
				b = Port B Address bit
				d = Port D Address bit
				D = data bit

When writing assembly code, Ra, Rb, and Rd are replaced by one of the actual register names, r0, r1, r2, … r7				

Table 1 – CPU Instruction Set Description

ECEN 240: Final Project Instruction Set and Decoder

Instruction Decoder PROM

The instruction decoder will receive a 4-bit instruction (opcode) from the program PROM (the 4 most significant bits), and configure the control signals of the functional blocks so that the instruction may be executed. The control signals consist of the enable signals and select lines listed in Table 3. To design the Instruction decoder, use Figure 1 to help you identify how each of the 7 functional blocks will be configured in order to execute each instruction. Enter these required signal states into the truth table.

This decoder could potentially be implemented by solving K-maps of the outputs of Table 3. Instead, you are asked to implement this logic with a Logisim ROM from the "Memory" menu. The ROM will have 4 address bits, and 7 data bits.
The required control bits are shown as outputs from the Instruction Decoder Prom in Figure 1. They are:

- Bit 0 = Register file write enable (WE).
- Bit 1 = The ASCII display write enable (DWE).
- Bits 3, 2 = The Register file Source Selector bits (RSS).
- Bits 4, 5 = ALU Function Selector bits (AFS).
- Bit 6 = Alters the addition function to subtraction (SUB).

Instruction Operation Code (opcode)		**Bit 6 ALU SUB**	**Bits 5, 4 ALU Function**	**Bits 3, 2 Register Source MUX**	**Bit 1 Display WE**	**Bit 0 Register WE**	**Combined Hex Equivalent**
`0000 [NOP]`		0	00	00	0	0	
`0001 [LD]`							
`0010 [MOV]`							
`0011 [DISP]`							
`0100 [XOR]`							
`0101 [AND]`							
`0110 [OR]`							
`0111 [ADD]`							
`1111 [SUB]`							

Table 3 – Instruction Decoder Truth Table

The left hand side of the truth table (the opcode) will be the address of each memory element of the instruction decoder PROM. The contents of the right hand side of the truth table will be enterred as data into the instruction decoder using hexadecimal numbers. Example: Address "0_{16}" of the PROM will contain "00_{16}" corresponding to the "No Operation" instruction.

Operation		Sub	S1	S0
A ^ B (XOR)		0	0	0
A & B (AND)		0	0	1
A | B (OR)		0	1	0
A + B (ADD)		0	1	1
A – B (Subtract)		1	1	1

Table 4 – ALU Operation Codes

ECEN 240: Final Project Instruction Set and Decoder

Programming Template

Assembly Code	Machine Code (Hex)	Description of Operation	Register Contents After Instruction Execution							
			r7	r6	r5	r4	r3	r2	r1	r0
LD r0, 0x3		Load “3” into r0								3
MOV r7, r0		Copy r0 contents (“3”) to r7	3							3