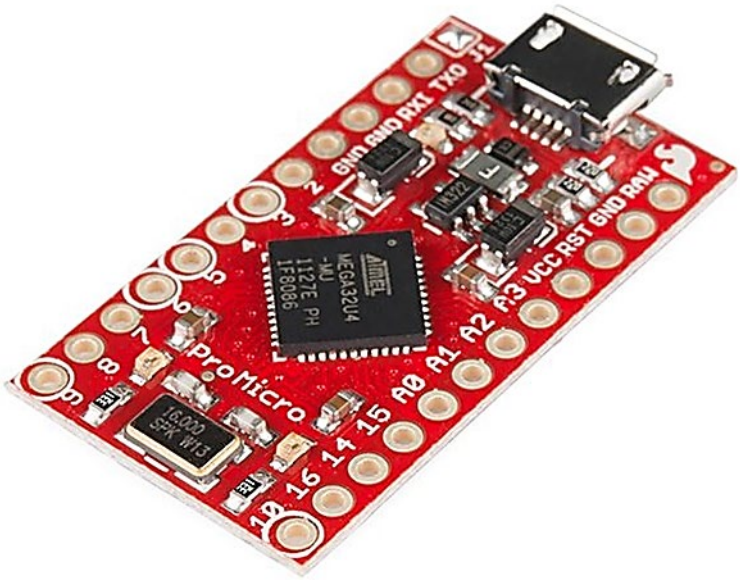
Electrical and Computer Engineering
Erik Jonsson School of Engineering & Computer Science
The University of Texas at Dallas
Dr. Tooraj Nikoubin



Project #3:
(1) Instruction Decoder
(2) Constant Unit

EEDG/CE 6302
Microprocessors and Embedded Systems

Objectives:

- Designing and simulating synthesizable ID (Instruction Decoder) and Constant Unit of the MCU
- Writing testbench to test the written codes

1. Introduction:

In this project, you will write HDL codes for other parts of the MCU, including ID and constant unit of the MCU. ID and PC (Program Counter) have to be simulated by your testbench.

2. Instruction Decoder

2.1. What is ID?

Also called Control Unit, ID makes required control signals to other parts of the MCU. These signals indicate how instructions should work, when registers should be read or written, when input port has to be read or output port has to be written, etc. If your ID is not designed correctly, your MCU won't function properly. Thus, you need to pay more attention to this part of the design and make sure it behaves as you want.

2.2. How it works?

In this project, ID gets 17 bits input from Instruction Register (IR) and decodes its contents. Any instruction can be categorized into three types: (1) Three-register type, (2) Two-register type and (3) Branch type.

Based on the type, an instruction has different parts. The leftmost fie bit of an instruction indicate the Opcode. Then, 3 bit address of destination register (DA) to where data will be written, 3 bit address of A source register (AA) from where data or address will be read. Other parts include: BB for address of B register, immediate value for those instructions that need direct values (for example add by constant value) and target address to memory for branch instructions. You can see these parts in the Figure (1). ID must decode any instructions and send the required control signals to other parts of the MCU. Note that ID is a combinational module and, hence, it does not contain any latch or flip-flop.

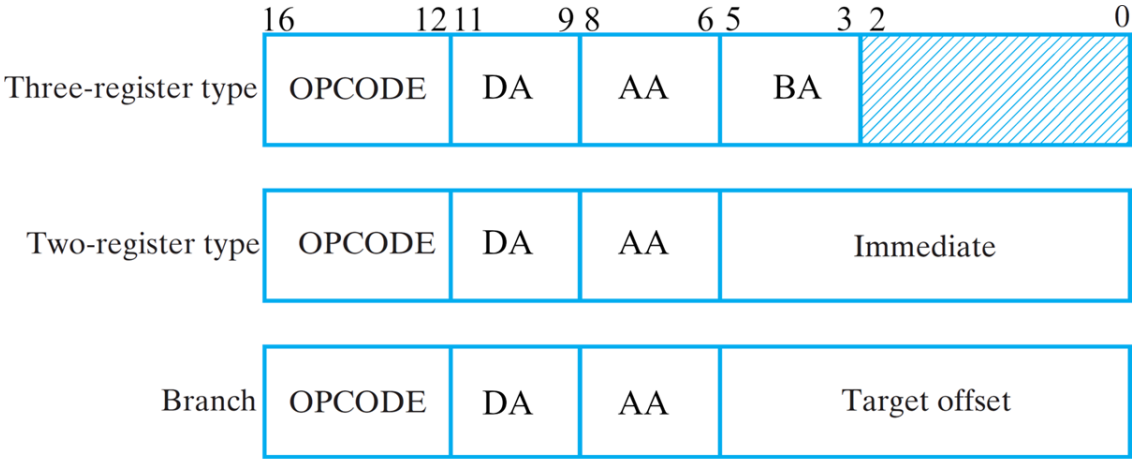


Figure (1): Different parts of an instruction

2.3. Implementation:

Look again to the block diagram of the MCU in Figure (2) at the end of this manual. As you can see, ID has only one input from the output of IR and fourteen output signals. Table (1) summarizes the function of each output.

	Width	Meaning
Instruction_in	17	Input instruction from IR
DA	3	Destination Address: address of the register to which data is written in the Register File
AA	3	A-register Address: address of the A-register from which data is read in the Register File
BA	3	A-register Address: address of the A-register from which data is read in the Register File
BS	2	Branch Select: detect branches and its effects on the Program Counter (PC)
PS	1	Zero Toggle: controls the type of conditional branch (Z or N)
MW	1	Memory Write: signal to the memory for enabling write; MW=='1' means write to the memory and MW=='0' means read from the memory
RW	1	Register Write: signal to the Register File for enabling write; RW=='1' means write to the specified register and RW=='0' means read from the specified register
MA	1	MUXA select bit
MB	1	MUXB select bit
MD	2	MUXD select bit
FS	4	ALU Function Select: specifies a function that ALU must perform
SH	3	Shift Number: Specifies the number of shifts in the ALU for related operations
CS	1	Constant Unit Select: controls zero fill or sign extension of immediate values in Constant Unit
OE	1	Output Enable: enables writing data to output port

Table (1): Input and outputs of Instruction Decoder

The logic that resides behind each output is summarized in the Table (2). In this table, 'don't care' is shown by 'X'. "Arb." In FS column means that you can decide which FS code belongs to each function in the ALU. Note that each "Arb." should be unique.

	RW	MD	BS	PS	MW	FS	MB	MA	CS	OE
NOP	0	XX	00	X	0	XXXX	X	X	X	0
MOVA	1	00	00	X	0	Arb.	X	0	X	0
ADD	1	00	00	X	0	Arb.	0	0	X	0
SUB	1	00	00	X	0	Arb.	0	0	X	0
AND	1	00	00	X	0	Arb.	0	0	X	0
OR	1	00	00	X	0	Arb.	0	0	X	0
XOR	1	00	00	X	0	Arb.	0	0	X	0
NOT	1	00	00	X	0	Arb.	X	0	X	0

ADI	1	00	00	X	0	Arb.	1	0	1	0
SBI	1	00	00	X	0	Arb.	1	0	1	0
ANI	1	00	00	X	0	Arb.	1	0	0	0
ORI	1	00	00	X	0	Arb.	1	0	0	0
XRI	1	00	00	X	0	Arb.	1	0	0	0
AIU	1	00	00	X	0	Arb.	1	0	0	0
SIU	1	00	00	X	0	Arb.	1	0	0	0
MOVB	1	00	00	X	0	Arb.	0	X	X	0
LSR	1	00	00	X	0	Arb.	X	0	X	0
LSL	1	00	00	X	0	Arb.	X	0	X	0
LD	1	01	00	X	0	XXXX	X	0	X	0
ST	0	XX	00	X	1	XXXX	0	0	X	0
JMR	0	XX	10	X	0	XXXX	X	0	X	0
SLT	1	10	00	X	0	Arb.	0	0	X	0
BZ	0	XX	01	0	0	Arb.	1	0	1	0
BNZ	0	XX	01	1	0	Arb.	1	0	1	0
JMP	0	XX	11	X	0	XXXX	1	X	1	0
JML	1	00	11	X	0	Arb.	1	1	1	0
IN	1	00	00	X	0	Arb.	0	0	0	0
INK	1	00	00	X	0	Arb.	0	0	0	0
OUT	0	00	00	X	0	XXXX	0	0	0	1

Other outputs, such as DA, AA or BA, can be directly extracted from any instruction and do not need any special logic. Also note that there are some extra logic gates at the output terminals of Instruction Decoder. For instance, BS goes to an AND gate, or BA goes to a comparator. Currently, you do not need to realize these logic parts in your HDL codes.
Three instructions are used to interface the MCU to the world: “IN” to read data, “INK” to read the value of the keyboard and “OUT” to write data to output port.

3. Constant Unit

3.1. What is CU?

You may notice that the immediate value in two-register-type instructions has 6 bits, but the data used in your ALU/FU requires 8 bits. To overcome this problem, we need to “extend” 6 bits to 8 bits. The duty of the Constant Unit in your MCU is to perform this extension.

3.2. Implementation

If CS==’0’, zero fills to the left of the immediate value. If CS==’1’, the sign extension should be performed by the Constant Unit on the immediate value.
Note that although the result of Constant Unit is used in certain instructions, it generates its output regardless of instructions. The output of the constant Unit will be connected to the MUXB input and, if needed, will be selected by the related control and logic signals.

4. Project requirements:

- Write a code in an HDL for Instruction Decoder and Constant Unit.
- Write a testbench for each code. Verify and show both your coding works correctly using waveforms. Your testbench should contain at least four

instructions for the Instruction Decoder. To show that the Constant Unit is working properly, consider all conditions in your testbench.

- Take a screenshot of the report of Xilinx ISE (or any other HDL synthesizers) which shows your Instruction Decoder and Constant Unit do not contain any latch or flip-flop and insert the screenshot in your report.
- Each student should submit a report no more than 4 pages (excluding cover page, TOC, appendix, etc).
- Do not include your codes in the report. Submit your HDL codes along with their testbenches separately.
- Answer to the questions at the end of this manual in the report.

5. Questions:

1. What is sign extension mean? Explain briefly in 1 to 2 lines.
2. Imagine that you want to add another instruction named “ADN". This specific instruction gets two inputs. It inverts one of them and add it with the other input. Into which category of instructions does “AND” fall?
3. If we want to have this instruction in our MCU, specify related control signals in the following table. (Do NOT add this instruction in your code.)

	RW	MD	BS	PS	MW	FS	MB	MA	CS	OE
ADN										

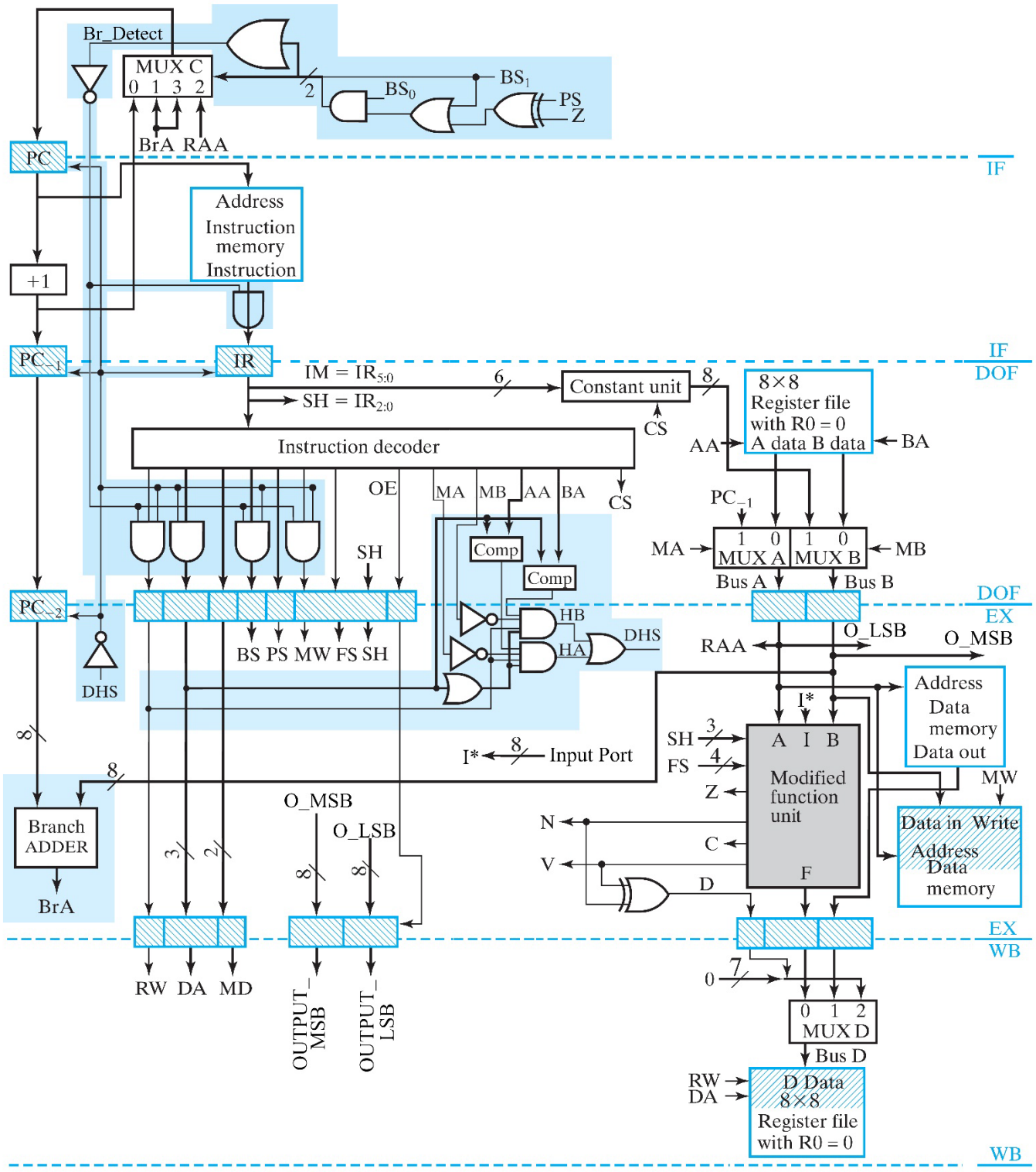


Figure (2): Block diagram of the MCU

Reference:

Mano, M. M. AND Kime, C. R. AND Martin M. "*Logic and Computer Design Fundamentals*", 5th ed. Hoboken, NJ: Pearson Higher Education Inc., 2015, pp 585-615.