Improved 8-Bit Processor (8BP2)

Table of Contents

Description

8BP2 is an 8-bit Van Neumann style processor that was designed for implementation using TTL logic. It is likely only the second of a long series of processors that were designed solely for the entertainment of the designer. Its small instruction set features a large set of arithmetic and logic instructions, I/O instructions, conditional branch instructions, as well as a few others. The processor also includes a single edge triggered maskable interrupt pin that can be expanded using an input port to determine the interrupt vector.

Pin Description

CLK – Clock Input.

ENABLE – Allows the processor to run when high. Warning: Not synchronized. Will need to be fixed.

!RESET – Resets processor when low.

D7-D0 – 8-bit data bus.

A15-A0 – 16-bit address bus which allows the processor to access up to 64KB.

INT – Triggers an interrupt when high. Multiple interrupt vectors can be added by connecting a priority encoder to an input port.

INTE – Enables interrupts when pin is high.

IJH – Pulled low by processor when interrupt is triggered to indicate that the data bus is floating and waiting to receive high byte of the address that the processor jumps to during interrupts. Can be connected to the output enable pin of a register in which the high byte of the address is stored.

IJL – Pulled low by processor when interrupt is triggered to indicate that the data bus is floating and waiting to receive low byte of the address that the processor jumps to during interrupts. Can be connected to the output enable pin of a register in which the low byte of the address is stored.

CO – Positive edge of CO indicates that the program counter (PC) has overflowed. (Note 10/27/2017: Does not work properly.)

Cn4 – Indicates that the ALU has carried. Has temporarily been removed from FPGA version to prevent a combinational loop.

IO7-IO0 – 8-bit I/O address that selects a port to read or write.

!IN – Pulled low by processor when an IN instruction is executed to read from selected input port.

OUT – Pulled high by processor when an OUT instruction is executed to write to an output port.

!RE – Pulled low by processor when reading from memory.

!WE – Negative edge of !WE latches the contents of the data bus into memory.

LPC – LPC should be connected to the clock pin of a two positive edge trigged 8-bit latches. The data inputs of the latches should be connected to PCVL and PCVH.

PCVL – Low byte of program counter. Should be connected to the data inputs of an 8-bit latch.

PCVH – High byte of program counter. Should be connected to the data inputs of an 8-bit latch.

Instruction Set and Assembly Language for 8BP2

Assembly Language

ALU

Format: ALU[f] a,b or ALU[f] a,b,c

The ALU operation can be entered as a number or as a symbol.

The set of symbols is:

+, Add a and b.

-, Subtract b from a.

++, Increment.

--, Decrement.

&, c equals a and b.

|, c equals a or b.

^, c equals a exor b.

~&, c equals a nand b.

~|, c equals a nor b.

~^, c equals a exnor b.

0, c equals zero.

-1, c equals twos complement negative one.

LPC

Format: LPC

IN

Format: IN[p] a

OUT
Format: OUT[p] a

TC
Format: TC a,b

JC
Format: JC[cc] a or JC[cc] a,b

RETI
Format: RETI

a,b,c: A variable or a constant that refers to a memory location.

f: A constant or symbol that tells the assembler which mathematical or logical operation is to be performed.

p: A constant number refers to a I/O port number.

cc: A constant or symbol that tells the assembler which condition that the processor should jump on.

Machine Code
X = Don't care. The instruction type should remain valid no matter what value is placed here.

ALU Instructions
Instruction format: 00XXXXXX, word 1 (address of A), word 2 (address of B), [word 3 (address of C)]?

Replace XXXXX in with the correct ALU code. To determine the correct code, refer to the table below or the 74*181 ALU datasheet. S3-S0 are connected to bits 3-0, M is connected to bit 4, and the *non-inverted* carry Cn is connected to bit 5 of the instruction register.

ALU Signal Connection

MSB							LSB
0	0	Cn	M	S3	S2	S1	S0

S0-S3, M Function select for 74181 ALU:

XXXXX	Store in	Function
00	B	A
01	C	A + B
02	C	A + !B
03	B	MINUS 1 (2's complement) A is used in this instruction to take up space.
04	C	A PLUS A!B
05	C	(A + B) PLUS A!B
06	C	A MINUS B MINUS 1

07	C	A!B MINUS 1
08	C	A PLUS AB
09	C	A PLUS B
0A	C	(A + !B) PLUS AB
0B	C	AB MINUS 1
0C	B	A PLUS A (Each bit is shifted to the next more significant position)
0D	C	(A + B) PLUS A
0E	C	(A + !B) PLUS A
0F	B	A MINUS 1
10	B	!A
11	C	!(A + B)
12	C	!AB
13	B	0 A is used in this instruction to take up space.
14	C	!(AB)
15	B	!B A is used in this instruction to take up space.
16	C	A ^ B
17	C	A!B
18	C	!A + B
19	C	!(A ^ B)
1A	B	B A is used in this instruction to take up space.
1B	C	AB
1C	B	1 A is used in this instruction to take up space.
1D	C	A + !B
1E	C	A + B
1F	B	A
20	B	A PLUS 1
21	C	(A + B) PLUS 1
22	C	(A + !B) PLUS 1
23	B	ZERO A is used in this instruction to take up space.
24	C	A PLUS A!B PLUS 1
25	C	(A + B) PLUS A!B PLUS 1
26	C	A MINUS B
27	C	A!B
28	C	A PLUS AB PLUS 1
29	C	A PLUS B PLUS 1
2A	C	(A + !B) PLUS AB PLUS 1
2B	C	AB
2C	B	A PLUS A PLUS 1
2D	C	(A + B) PLUS A PLUS 1
2E	C	(A + !B) PLUS A PLUS 1
2F	B	A
30	B	!A
31	C	!(A + B)
32	C	!AB
33	B	0 A is used in this instruction to take up space.
34	C	!(AB)
35	B	!B A is used in this instruction to take up space.
36	C	A ^ B

37	C	A!B	
38	C	!A + B	
39	C	!(A ^ B)	
3A	B	B	A is used in this instruction to take up space.
3B	C	AB	
3C	B	1	A is used in this instruction to take up space.
3D	C	A + !B	
3E	C	A + B	
3F	B	A	

TC

Instruction format: 010XXXXX, word 1 (address of A), word 2 (address of B)

Compares the magnitude of bytes 1 and 2 and stores that information in the condition code register.

LPC

Instruction format: 011XXXXX

Pulses LPC pin in order to store the program counter in an input port latch. The processor can potentially be wired to send the pulse to another device. For example, LPC is connected to an acknowledge line on some peripheral.

```
wire LPC;

wire ACK;

assign ACK = LPC;
```

IN

Instruction format: 100XXXXX, byte 1 (port), word 2 (address of A)

Sends the byte stored at the address of word 2 to the port pointed to by byte 1.

OUT

Instruction format: 101XXXXX, byte 1 (port), word 2 (address of A)

Stores the byte in the port pointed to by byte 1 in the address of word 2.

RETI

Instruction format: 110XXXXX

The program counter is loaded with the previous location in the program after an interrupt.

JC

Instruction format: 111XXXXX, word 1 (address of high byte), word 2 (address of low byte)

The processor checks the condition codes, and if the CCs match the selected condition the processor jumps to the location of the word that bytes 1 and 2 point to.

Bits 2-0 of the instruction determine the condition to jump on.

Bits 2-0	condition
0	don't jump
1	jump if A=B
2	jump if A<B
3	jump if A≤B
4	jump if A>B
5	jump if A≥B
6	jump if A≠B
7	jump

Processor Operation

ALU

The ALU uses the 74181 48-function ALU which also provides a carry and !(A=B) output that connect directly to the condition code register. The A input is attached to the ALU through a register while the B input is connected directly to the data bus. The F output is connected to the ALU and the data bus using a three-state register.

Program Counter

The program counter is 16 bits and is loaded by storing the high byte in a register and then placing the low byte onto the data bus. The control logic then loads the contents of the register and the data bus into the program counter. If the program counter overflows a flip-flop is set that can turn on an error light or trigger an interrupt.

Address Registers

There are four address registers that have two uses. Two are loaded from the data bus to address memory without using the program counter while the other two store the last value of the program counter before an interrupt happened.

I/O Register

The I/O Register stores the address of the input or output port to be written or read from.

Interrupt Logic

When the INT pin rises from low to high and the INTE pin is high as well, the program counter is loaded into two of the address registers and the program counter is loaded with two bytes that should be placed onto the data bus when IJH or IJL are high. To return from an interrupt the RETI instruction is executed.

Control Logic

At the start of each instruction cycle a byte is taken from memory and loaded into the instruction register. The actions that the processor takes from then to the end of the cycle are determined by the instruction register.

If the instructions do not satisfy you, you can reprogram most of the microcode to change the instructions or even to expand the instruction set.