Simple 32-Bit Processor

Clemson Cadence Summer Project Proposal

JB Bahrenburg

jbahren@clemson.edu

Clemson University

July 11, 2023

Introduction:

This 32-bit computer processor will include an instruction set consisting of the 34 functions listed in the table below. Each instruction is listed with its corresponding opcode and a description of the function's purpose. This processor is designed to interact with a RAM main memory system as well as a stack. This means the processor will use two special purpose registers: a program counter and a stack pointer. Along with these special purpose registers, the processor will have 30 general purpose registers, all of which connect to a 32-to-1 multiplexer. Some of the other special purpose registers used include an instruction register, operand register, and accumulator. The ALU will perform several arithmetic functions: Add, subtract, multiply, divide, AND, OR, XOR, and NOT. The other arithmetic functions done are shift functions, which will be done by a shift register.

Instructions received by the processor will be 32-bits wide and will come in one of three formats. In each format the first 6 bits will be the opcode, which will also determine which format the instruction is in. The first format is the "arithmetic format", used for performing certain arithmetic operations such as adds and shifts. After the 6-bit opcode, the next 15 bits point to the registers which will be used for the operation, including two source registers, and one destination register. The next 5 bits are used to determine how many bits to shift in the case of a shift instruction, The final 6 bits are unused. A diagram of the format can be seen below.

Opcode = 000000	RS	RT	RD	Sh.Amt.	
6 bits	5	5	5	5	6

Figure 1: Arithmetic instruction format. [1]

The second format, "immediate" handles mostly immediate functions, as well as loads and stores. Immediate functions include adds and subtracts in which the "immediate" value stored in the instruction register is, for example, added to a register value (instead of two register values being added). After the 6-bit opcode, the next 10 bits are used to point to two general purpose registers. The final 16 bits are used for an address when needed. The format can be seen below in figure 2.

Opcode	RS	RD	Immediate/Address
6 bits	5 bits	5 bits	16 bits

Figure 2: Immediate instruction format. [1]

The final format is the jump format and handles any of the jump-related functions. This includes the call and return functions. Since these functions really only rely on addresses, the format only uses 6 bits for the opcode and 26 bits for the address needed by the function, as seen below in figure 3.

Opcode	Address
6 bits	26 bits

Figure 3: Jump instruction format. [1]

Opcode	**Function**	**Bit Format**	**Description**

000000	Load	Immediate	Load the value to register A stored in the address pointed to by register B
000001	Store	Immediate	Store the value from register A to address pointed to by register B
000010	Move	Immediate	Move the contents of register A to register B, clear register A
000011	Copy	Immediate	Copy the contents of register A into register B
000100	Add	Arithmetic	Add the contents of register A to the contents of register B, store in register C
000101	Subtract	Arithmetic	Subtract the contents of register A from the contents of register B, store in register C
000110	AND	Arithmetic	AND the contents of register A to the contents of register B, store in register C
000111	OR	Arithmetic	OR the contents of register A to the contents of register B, store in register C
001000	XOR	Arithmetic	XOR the contents of register A to the contents of register B, store in register C
001001	NOT	Immediate	NOT the contents of register A, store in register B
001010	Immediate Load	Immediate	Load the lower 16 bits (integer) of the instruction register, sign extend and store in register A
001011	Immediate Add	Immediate	Add the contents of the instruction register to the contents of register A, store in register B
001100	Immediate Subtract	Immediate	Subtract the contents of the instruction register from register A, store in register B
001101	Immediate AND	Immediate	AND the contents of the instruction register to the contents of register A, store in register B
001110	Immediate OR	Immediate	OR the contents of the instruction register to the contents of register A, store in register B
001111	Immediate XOR	Immediate	XOR the contents of the instruction register to the contents of register A, store in register B

010000	Immediate NOT	Immediate	NOT the contents of the instruction register, store in register A
010001	Logical Shift Left	Arithmetic	Shift the contents of register A *shamt* bits, store in register B
010010	Logical Shift Right	Arithmetic	Shift the contents of register A *shamt* bits, store in register B and replace vacated bits with 0s
010011	Arithmetic Shift Right	Arithmetic	Shift the contents of register A *shamt* bits, store in register B and replace vacated bits with the sign bit
010100	Jump	Jump	Go to specified address
010101	Compare	Immediate	Subtract the value of register A from register B
010110	Jump if Zero (Jump if equal)	Jump	If the compare result is 0, jump
010111	Jump if not zero (jump if not equal)	Jump	If the compare result is not 0, jump
011000	Jump if greater than	Jump	Jump if the compare result is positive
011001	Jump if greater than or equal	Jump	Jump if the compare result is 0 or positive
011010	Jump if less than	Jump	Jump if the compare result is negative
011011	Jump if less than or equal	Jump	Jump if the compare result is negative or 0
011100	Two's complement negation	Immediate	Perform a two's complement negation of register A
011101	Increment	Immediate	Increment the contents of register A by 1
011110	Double	Immediate	Double the value of the contents of register A, (shift left 1)
011111	Push	Immediate	Push the value stored in register A onto the top

			of the stack
100000	Divide	Arithmetic	Divide the contents of register A by the contents of register B, store in register C
100001	Call	Jump	Pop the address from the top of the stack and jump to that address, push the previous instruction onto the stack
100010	Return	Jump	Pop the address (from the call instruction) from the top of the stack and jump to the address
100011	Pop	Immediate	Pop the address from the top of the stack and store in register A
100100	Immediate Divide	Immediate	Divide the contents of register A by the contents of the instruction register, store in register B

References

1. https://staffwww.fullcoll.edu/aclifton/cs241/lecture-instruction-format.html
2. https://www.youtube.com/watch?v=1bP6alXjDrw