Design Document of I-Mode Memory Instructions Extension

Author: Xinyu Kang UID: u6120911 Date: 26/04/2022

Overview

Starting from the CPU implemented in labs, I extended the QuAC ISA, making it be able to run **I-Mode Memory Instructions**. The two extended instructions: **sti** and **ldi**, *allow a store/load directly from a specified memory location*. For example:

```
sti r1, 0xA  ; Stores the word held by r1 into memory at address 0xA
ldi r4, 0xC  ; Reads the word in memory at address 0xC and writes to r4
```

(more sample code -- ***sample_code.asm*** in root folder)

By using the instruction **sti** and **ldi**, the 8-bit immediate can be easily accessed, avoiding loading addresses into registers first. This is especially useful given the limited number of general-purpose registers available.

The extension filled the gap of two undefined I-Mode operands in QuAC ISA. Now, there are four **I-Mode instructions**:

Syntax	**Semantic**	**Machine Code**
movl{z} rd, imm8	rd := #imm8	0000 z<rd> <imm8>
seth{z} rd, imm8	rd =(#imm8 << 8)|(rd & 0xff)	0001 z<rd> <imm8>
sti{z} rd, imm8	mem[imm8]:= rd	0010 z<rd> <imm8>
ldi{z} rd, imm8	rd := mem[imm8]	0011 z<rd> <imm8>

Implementation

a. Control Unit

First, the control unit needs to identify the **sti** and **ldi** instructions. In my implementation, bit 12-15 of the machine code is connected to a demultiplexer, generating 16 instruction signals (6 undefined). If bit 12-15 is 0010, then it's a **sti** instruction, and if bit 12-15 is 0011 then it's a **ldi** instruction. Then, modify the control signals so that rest of the CPU can execute the instruction correctly.

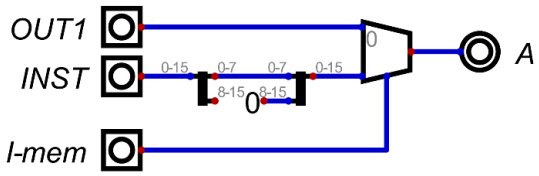
	RS1	RS2	WE	WS	STR	LDR	DMUX
sti	--	rd	0	--	1	0	11
ldi	--	--	1	rd	0	1	10

(signals STR and LDR can also be named as ST and LD, because they connect to the memory no matter in R-mode or I-mode)

Last, output a new signal, **I-mem**, indicating whether it is an I-Mode memory instruction. This signal will be used in modifying the memory's address input.

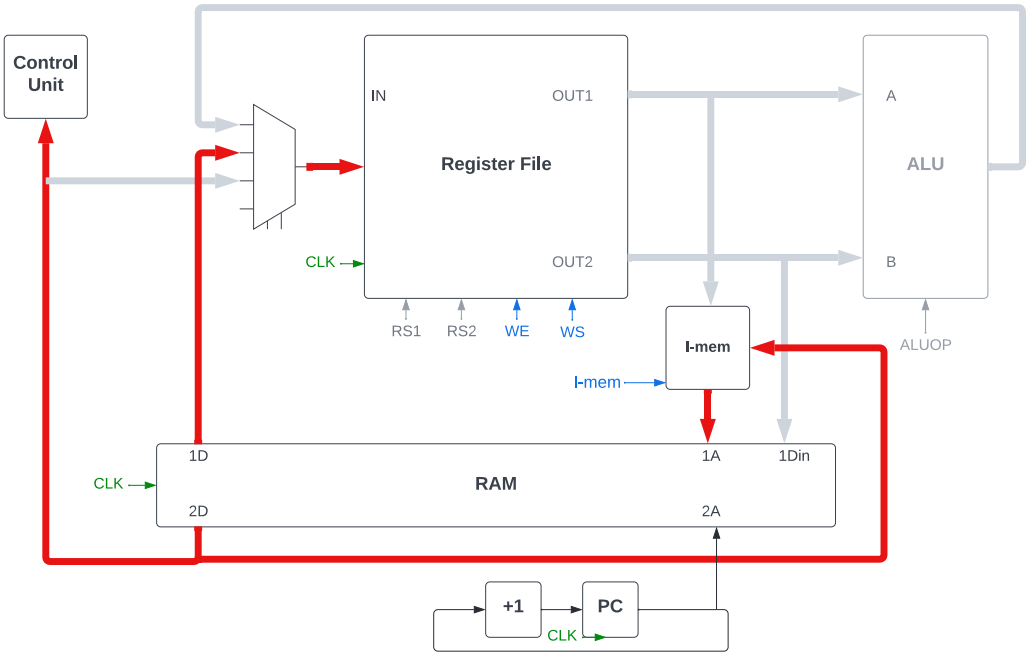
b. Memory's Address Input

In R-mode, memory's address input is connected to OUT1 of the register file, which is the value read from the selected register. Now, I want it to directly connect to the instruction machine code INST and use the first 8 bits of the code as the address to store/load. So, the following circuit is created, using **I-mem** signal to decide the memory's address input should come from the register or the instruction.



c. Instruction Behavior

The instruction behavior is almost same as **str**/**ldr**, except that the memory address comes from the instruction instead of the register. Here's a flow chart of **ldi**:



Reflection

By implementing this extension, I have got the general idea of how to work with the immediate in QuAC ISA. There is a key limitation that the instruction can only carry an 8-bit address while in R-mode the register can store a 16-bit address. Also, it might be a little wasteful to define two new instructions to implement similar features with existing instructions. An initial thought was to use the z-bit of the str/ldr instruction to determine the mode, if it's in I-mode then treat 0-8 bits as the imm8. However, any extensions must be backwards compatible with the base ISA. The condition execution should not be modified, so I use this design method instead.