CSE 151A HW 01

Karlo Godfrey Escalona Gregorio
PID: A16536865

Contents

1 Preface

This project explores a custom implementation of a subset of the ARM instruction set. A custom instruction set inspired by the ARM architecture is designed with a custom assembler. The architecture is implemented in hardware as an RTL model, whose functionality is verified

The assembler is implemented in Python, and the RTL model is implemented using SystemVerilog.

It should be noted that this architecture is an educational project inspired by ARM-style RISC design using the ARM7TDMI-S data sheet as refereence. It is not ARM-compatible and does not use proprietary ARM encoding or IP.

2 ISA Design

All instruction words are designed to be 32 bits wide. Each instruction has 4 condition bits that will determine whether or not the instruction executes based on CPSR condition flags (N, Z, C, V). This makes it simpler to write conditional statements for simple instructions. A list of the condition codes is listed below.

Field List			
Condition Code	**Instruction Suffix**	**Flags Set (NZCV)**	**Explanation**
0000	unused	N/A	unused
0001	always	flags ignored	Always Executed
0010	LE	Z set OR (N not equal to V)	Less Than or Equal
0011	GT	Z clear AND (N equals V)	Greater Than
0100	LT	N not equal to V	Less Than
0101	GE	N equals V	Greater Or Equal
0110	LS	C clear or Z set	Unsigned Lower or Same
0111	HI	C set and Z clear	Unsigned Higher
1000	VC	V clear	No Overflow
1001	VS	V set	Overflow
1010	PL	N clear	Positive or Zero
1011	MI	N set	Negative
1100	CC	C clear	Unsigned Lower
1101	CS	C set	Unsigned Higher or Equal
1110	NEQ	Z clear	Not Equal
1111	EQ	Z set	Equal

2.1 RX-type

The RX-type is used for fixed-point arithmetic data-processing instructions. A summary of the format can be seen in Figure 1, and explanations of the fields can be seen under the figure.

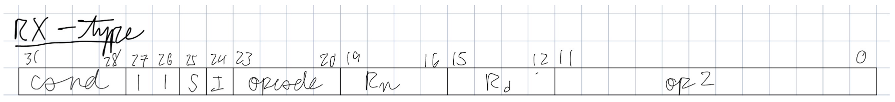

Figure 1: RX instruction type format.

Field List		
Field	**Bits**	**Description**
cond	[31:28]	State of CPSR condition codes (based on NZCV flags)
type	[27:26]	Encoding specific to instruction type
S	25	Determines whether or not to alter condition codes (S = 0 means do not alter)
I	24	Determines whether or not op2 is an immediate (I = 0 means op2 is not an immediate, but a shift register)
opcode	[23:20]	Determines the operation performed on operands
R_n	[19:16]	First source register
R_d	[15:12]	Destination register
$op2$	[11:0]	Varying field depending on the value of opcode

RX-type instructions have a varying $op2$ field that can be used depending on whether or not the instruction uses an immediate. A summary of the $op2$ fields is shown below.

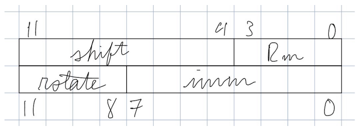

Figure 2: RX instruction type format.

Field List		
Field	**Bits**	**Description**
shift	[11:4]	Used for instructions using two source registers. The amount to shift the value in R_m
R_m	[3:0]	Used for instructions using two source registers. The second source register
rotate	[11:8]	Used for instructions using one source register and one immediate. Rotates the immediate a specific number of positions
imm	[7:0]	A constant used with another shift register to produce the result

A list of suported instructions is listed below.

Field List	
Field	**Description**
add.x	Adds two fixed-point values.
sub.x	[3:0]
mul.x	[11:8]
div.x	[7:0]
addi.x	[7:0]
subi.x	[7:0]
mac.x	[7:0]
sqrt.x	[7:0]
convf.x	[7:0]

2.2 RF-type

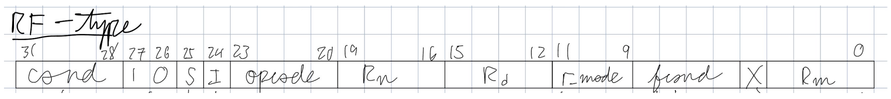

Figure 3: RF instruction type format.

2.3 D-type

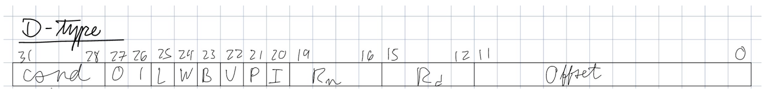

Figure 4: D instruction type format.

2.4 B-type

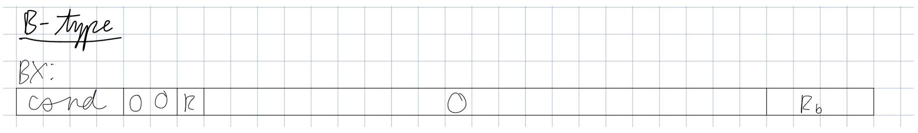

Figure 5: B instruction type format for BX instruction

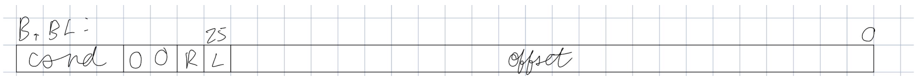

Figure 6: B instruction type format for B and BL instruction