The BAW Instruction Set Manual

Bryant Herren, Wesley Ring, Austin Waddell

Contents

1 Introduction

BAW is the instruction set for the processor we are building for ECGR 3183. It is a floating point co-processor with a single-cycle architecture, and a pipelined architecture. For the pipelined architecture, we implement two branch prediction algorithms (1 static and 1 dynamic) as well as no branch prediction.

We will be providing:

- the documented ISA
- architecture and controller design (units, diagrams, etc)
- VHDL/C++/other simulation
- performance results and discussion for pipelined vs. unpipelined approaches

1.1 System Parameters

- 32 Bits
- Data is stored using IEEE single-precision floating point numbers.
- There are 16 Registers
- Timings:
  - Clock cycle (pipelined): 100ns
  - Register Read/Write: 100ns
  - Memory Read/Write: 300ns
  - Single ALU Op: 200ns

1.2 Memory

The system includes a data memory addressed 0-1023 and 16 Floating Point registers (Referenced as X0-X15). Each memory location and register uses a 32-bit value. The simulation can read an input file containing the operational parameters, code, and memory contents (there is an assembler).

1.3 Additional Features

- FP Multiply by -1, 1, or 0 takes 1 cycle
- FP Multiply by power of 2 takes 2 cycles
- Condition Codes: ZNV (zero, negative, overflow)
  - All condition codes are set as needed on arithmetic operations (pay special attention to the FP ALU operations and results)
- The round, ceiling, and floor functions would round up to the nearest integer (not always a power of 2), expressing the result in floating point format.

2 ISA

2.1 Introduction

Our ISA is based off of the ARMv8 / LEGv8. Because of this, you will likely see similarity.

2.2 Instruction Format

We have three instruction formats, Register (R), Data (D), and Immediate (I). All processor instructions are 32 bits wide. Table 1 Contains information about the specific instruction formats.
Notes:

1. The opcode is 8 bits long, allowing to be easily read in hex format.
2. The remaining three bits of each instruction format are not used.

Please see Table 2 for specific information on the Instruction Formats.

Table 1: About the formats

Format	Description	Example
R (Register)	An instruction whose inputs and outputs are both registers	ADD X9, X21, X9
D (Data)	Used when fetching or placing data in memory	LOAD X9, [X22, #64]
I (Immediate)	An instruction with data in the instruction	POW X9, #15

Table 2: Instruction Formats

Instruction Format	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
R (Register)	opcode (8 bits)								Rm (5 bit)					Empty (7 bit)							Rn (5 bit)					Rd (5 bit)					Empty		
D (Data)	opcode (8 bits)								Address (9 bit)										op2 (2 bit)		Rn (5 bit)					Rd (5 bit)							
I (Immediate)	opcode (8 bits)								Immediate Data (11 bit)												Rn (5 bit)					Rd (5 bit)					(3 bit)		

2.3 Instructions

2.3.1 Set

SET

ASM	opcode	Format	Description	Operation	ALU Cycles
SET	00001	I	Sets Ri to given floating point value	Ri $\leftarrow$ FPvalue	1

ASM Example: Set Ri, #FPvalue

Flags
- Zero
- Negitive
- Error

2.3.2 Load

LOD

ASM	opcode	Format	Description	Operation	ALU Cycles
LOD	00010	D	Copies Rj from memory and into Ri	Ri $\leftarrow$ M[Rj]	1

ASM Example: Load Ri, Rj

Flags
- Zero
- Negitive
- Error

2.3.3 Store

STR

ASM	opcode	Format	Description	Operation	ALU Cycles
STR	00011	D	Copies data from register Rj into memory	M[Ri] $\leftarrow$ Rj	1

ASM Example: Store Ri, Rj

Flags
- Zero
- Negitive
- Error

2.3.4 Move

MOV

ASM	opcode	Format	Description	Operation	ALU Cycles
MOV	00100	R	Moves the value of Rj to Ri, deleting the original	Ri $\leftarrow$ Rj	1

ASM Example: Move Ri, Rj

Flags
- Zero
- Negitive
- Error

2.3.5 Add

ADD

ASM	opcode	Format	Description	Operation	ALU Cycles
ADD	00101	R	Adds Rj and Rk into Ri	Ri ← Rj + Rk	3

ASM Example: Fadd Ri, Rj, Rk

Flags
- Zero
- Negitive
- Error

2.3.6 Subtract

SUB

ASM	opcode	Format	Description	Operation	ALU Cycles
SUB	00110	R	Subtrcts Rk from Rj into Ri	Ri ← Rj − Rk	3

ASM Example: Fsub Ri, Rj, Rk

Flags
- Zero
- Negitive
- Error

2.3.7 Negate

NEG

ASM	opcode	Format	Description	Operation	ALU Cycles
NEG	00111	R	Sets Ri to the Opposite of Rj	Ri ← -Rj	1

ASM Example: Fneg Ri, Rj

Flags
- Zero
- Negitive
- Error

MUL

ASM	opcode	Format	Description	Operation	ALU Cycles
MUL	01000	R	Multiplies Rj and Rk into Ri	Ri ← Rj * Rk	5

2.3.8 Multiply

ASM Example: Fmul Ri, Rj, Rk

Flags

- Zero
- Negitive
- Error

2.3.9 Divide

DIV

ASM	opcode	Format	Description	Operation	ALU Cycles
DIV	01001	R	Divides Rj by Rk into Ri	Ri ← Rj / Rk	8

ASM Example: Fdiv Ri, Rj, Rk

Flags

- Zero
- Negitive
- Error

2.3.10 Floor

FLR

ASM	opcode	Format	Description	Operation	ALU Cycles
FLR	01010	R	Sets Ri to the floor of Rj	Ri ← Rj	1

ASM Example: Floor Ri, Rj

Flags

- Zero
- Negitive
- Error

2.3.11 Ceiling

CEL

ASM	opcode	Format	Description	Operation	ALU Cycles
CEL	01011	R	Seting Ri to the ceil of Rj	Ri ← Rj	1

ASM Example: Ceil Ri, Rj

Flags

- Zero
- Negitive
- Error

2.3.12 Round

RND

ASM	opcode	Format	Description	Operation	ALU Cycles
RND	01100	R	Sets Ri to Rj rounded to the nearest whole number	Ri ← round(Rj)	1

ASM Example: Round Ri, Rj

Flags

- Zero
- Negitive
- Error

2.3.13 Absolute Value

ABS

ASM	opcode	Format	Description	Operation	ALU Cycles
ABS	01101	R	Sets Ri to the absolute value of Rj	Ri ← | Rj |	1

ASM Example: Fabs Ri, Rj

Flags

- Zero
- Negitive
- Error

2.3.14 Minimum

MIN

ASM	opcode	Format	Description	Operation	ALU Cycles
MIN	01110	R	Sets Ri to the minimum value between Rj and Rk?	Ri ← min(Rj, Rk)	1

ASM Example: Min Ri, Rj, Rk

Flags

- Zero
- Negitive
- Error

2.3.15 Maximum

ASM Example: Max Ri, Rj, Rk

Flags

- Zero
- Negitive
- Error

MAX

ASM	opcode	Format	Description	Operation	ALU Cycles
MAX	01111	R	Sets Ri to the maximum value between Rj and Rk?	Ri ← max(Rj, Rk)	1

2.3.16 Power

POW

ASM	opcode	Format	Description	Operation	ALU Cycles
POW	10000	I	Sets Ri to Rj raised to some given integer power	Ri ← Rj^integer_value	6

ASM Example: Pow Ri, Rj, #integer_value

Flags
- Zero
- Negitive
- Error

2.3.17 Exponent

EXP

ASM	opcode	Format	Description	Operation	ALU Cycles
EXP	10001	R	Sets Ri to Rj exponentiated	Ri ← e^Rj	8

ASM Example: Exp Ri, Rj

Flags
- Zero
- Negitive
- Error

2.3.18 Square Root

SQR

ASM	opcode	Format	Description	Operation	ALU Cycles
SQR	10010	R	Sets Ri to the square root of Rj	Ri ← $\sqrt{}$ Rj	8

ASM Example: Sqrt Ri, Rj

Flags
- Zero
- Negitive
- Error

BRU

ASM	opcode	Format	Description	Operation	ALU Cycles
BRU	10011		Loads Ri from memory into PC	PC ← M[Ri]	1

2.3.19 Branch (Uncond.)

ASM Example: B Ri

Flags
- Zero
- Negitive
- Error

2.3.20 Branch Zero

BRZ

ASM	opcode	Format	Description	Operation	ALU Cycles
BRZ	10100		Sends the PC to a specific labeled line if Ri is zero	If (Ri == 0) PC ← LABEL (line)	3

ASM Example: BZ Ri, LABEL

Flags
- Zero
- Negitive
- Error

2.3.21 Branch Negative

BRN

ASM	opcode	Format	Description	Operation	ALU Cycles
BRN	10101		Sends the PC to a specific labeled line if Ri is negative	If (Ri < 0) PC ← LABEL (line)	3

ASM Example: BN Ri, LABEL

Flags
- Zero
- Negitive
- Error

2.3.22 No-op

NOP

ASM	opcode	Format	Description	Operation	ALU Cycles
NOP	10110		No operation	No operation	1

ASM Example: Nop

Flags

- Zero
- Negitive
- Error

2.3.23 Halt

HLT

ASM	opcode	Format	Description	Operation	ALU Cycles
HLT	10111		Stop program	Stop Program	-

ASM Example: Halt

Flags

- Zero
- Negitive
- Error

3 Architecture

3.1 Datapath

3.2 controller

4 VHDL Description

5 Testing

6 Conclusion