Project:

- To develop a micro RISC ISA processor for IoT and related applications.
- Development includes:
    - Assembler
    - RTL
    - Test bench
    - Test suites including self-checking functional tests
    - Stretch Goal: JTAG debugger
    - Micro architecture and user documents
- Demonstration in Analog Devices simulation environment
- This development is only for educational purpose should not be used for commercial purpose
- Work should not be shared without the permission of Analog Devices
- Analog Devices reserves the right to use this work

Page: 447 of:

https://www.analog.com/media/en/dsp-documentation/processor-manuals/50836807228561ADSP2106xSHARCProcessorUsersManual_Revision2_1.pdf

Architecture:

Data registers/datapath = 16bit

Address = 16 bit

Number of

Types:

3a. IF condition compute; //detailed list of compute given later

IF condition DM(Ia,Mb) = ureg ;

3c. IF condition compute;

IF condition ureg = DM(Ia,Mb) ;

5. IF condition compute, ureg1 = ureg2;

7 (modified). IF condition MODIFY (Ia,Mb) ;

9. IF condition JUMP (Md,Ic);

17 (modified for data length = 16) ureg = <data16>;

20 (modified). PUSH PCSTK = <ureg>;

<ureg> = POP PCSTK;

(Comment: PCSTK is Return address stack, this is modified to make complete operation as one instruction)

21. NOP;

22. IDLE (optional)

Conditions (A-5):

Condition Description

EQ ALU equal zero

LT ALU less than zero

LE ALU less than or equal zero

AC ALU carry AV ALU overflow

MV Multiplier overflow

MS Multiplier sign

SV Shifter overflow

SZ Shifter zero

NE ALU not equal to zero

GE ALU greater than or equal zero

GT ALU greater than zero

NOT AC Not ALU carry

NOT AV Not ALU overflow

NOT MV Not multiplier overflow

NOT MS Not multiplier sign

NOT SV Not shifter overflow

NOT SZ Not shifter zero

FOREVER Always True

Registers (page: A-6):

Note: All registers are 16bit

Data Register File

R15 - R0 Register file locations, fixed-point

Program Sequencer

PC Program counter (read-only)

PCSTK Top of PC stack

PCSTKP PC stack pointer

Data Address Generators

I7 - I0 DAG1 index registers

M7 - M0 DAG1 modify registers

L7 - L0 DAG1 length registers

B7 - B0 DAG1 base registers

Timer

TPERIOD Timer period

TCOUNT Timer counter (As soon as non-zero value is written, timer will start)

Map 1 Registers:

PC	program counter
PCSTK	top of PC stack
PCSTKP	PC stack pointer
FADDR	fetch address
DADDR	decode address
LADDR	loop termination address
CURLCNTR	current loop counter
LCNTR	loop counter
R15 - R0	register file locations
I15 - I0	DAG1 and DAG2 index registers
M15 - M0	DAG1 and DAG2 modify registers
L15 - L0	DAG1 and DAG2 length registers
B15 - B0	DAG1 and DAG2 base registers

System Registers:

MODE1	mode control 1
MODE2	mode control 2
IRPTL	interrupt latch
IMASK	interrupt mask
IMASKP	interrupt mask pointer
ASTAT	arithmetic status
STKY	sticky status
USTAT1	user status reg 1
USTAT2	user status reg 2

System Registers

(b7=0)

b7 b6 b5 b4

b3 b2 b1 b0	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1
0 0 0 0	R0	I0	M0	L0	B0		FADDR	USTAT1
0 0 0 1	R1	I1	M1	L1	B1		DADDR	USTAT2
0 0 1 0	R2	I2	M2	L2	B2			
0 0 1 1	R3	I3	M3	L3	B3		PC	
0 1 0 0	R4	I4	M4	L4	B4		PCSTK	
0 1 0 1	R5	I5	M5	L5	B5		PCSTKP	
0 1 1 0	R6	I6	M6	L6	B6		LADDR	
0 1 1 1	R7	I7	M7	L7	B7		CURLCNTR	
1 0 0 0	R8	I8	M8	L8	B8		LCNTR	
1 0 0 1	R9	I9	M9	L9	B9			IRPTL
1 0 1 0	R10	I10	M10	L10	B10			MODE2
1 0 1 1	R11	I11	M11	L11	B11			MODE1
1 1 0 0	R12	I12	M12	L12	B12			ASTAT
1 1 0 1	R13	I13	M13	L13	B13			IMASK
1 1 1 0	R14	I14	M14	L14	B14			STKY
1 1 1 1	R15	I15	M15	L15	B15			IMASKP

Computes (*Page: 504*):

Fixed point computes involve 16 bit sign extended integer oprands in all cases

B.2.1 ALU Operations

The ALU operations are described in this section. Tables B.1 and B.2 summarize the syntax and opcodes for the fixed-point and floating-point ALU operations, respectively. The rest of this section contains detailed descriptions of each operation.

Syntax Opcode

Rn = Rx + Ry

Rn = Rx – Ry

Rn = Rx + Ry + CI

Rn = Rx – Ry + CI – 1

COMP(Rx, Ry)

Rn = –Rx

Rn = ABS Rx

Rn = Rx AND Ry //bit by bit

Rn = Rx OR Ry //bit by bit

Rn = Rx XOR Ry //bit by bit

Rn = REG_OR Rx // OR all the bits of Rx - New

Rn = REG_AND Rx // AND all the bits of Rx - New

Rn = NOT Rx

Rn = MIN(Rx, Ry)

Rn = MAX(Rx, Ry)

B.2.2 Multiplier Operations

The multiplier operations are described in this section. Table B.3 summarizes the syntax and opcodes for the fixed-point and floating-point multiplier operations. The rest of this section contains detailed descriptions of each operation.

MRF is a40 bit accumulate register made by joining (MR2[7:0]+MR1[15:0]+MR0[15:0]) registers

Rn = Rx * Ry

MRF = Rx * Ry

Rn = MRF + Rx * Ry

MRF = MRF + Rx * Ry

Rn = MRF – Rx * Ry

MRF = MRF – Rx * Ry

Rn = SAT MRF

B.2.3 Shifter Operations (page: 556)

Rn = ASHIFT Rx BY Ry //Positive value of *Ry* will do left shift, negative will do right shift on *Rx, page 560*

Rn = ROT Rx BY RY //Positive value of *Ry* will do left shift, negative will do right shift on *Rx, page 561*

Rn = LEFTZ Rx //Function: Extracts the number of leading 0s from the fixed-point operand in Rx.

Rn = LEFTO Rx //Function: Extracts the number of leading 1s from the fixed-point operand in Rx.