Eng. Julian S. Bruno

REAL TIME DIGITAL SIGNAL PROCESSING

UTN-FRBA
2010

Eng. Julian Bruno

Architecture

Introduction to the Blackfin Processor

Eng. Julian S. Bruno

Introduction to MSA

- Micro Signal Architecture (MSA) core was jointly developed by Intel and Analog Devices Inc (ADI).

- MSA incorporates both DSP and microcontroller functionalities in a single core.

- MSA include optimizations for high-level language programming, memory protection, and byte addressing.

- MSA has a very efficient and dynamic power management feature.

- Adjuts boyh the voltage delivered to the core and the frequency at which core runs.

Eng. Julian S. Bruno

Blackfin Processor

- The BF processor is based on the MSA core.
- 16-/32-bit embedded processor core with a 10-stage RISC MCU/DSP pipeline
- Dual MAC signal processing.
- Flexible Single Instruction.
- Multiple Data (SIMD) capabilities.
- Multimedia processing features into a single instruction set architecture.
- Instruction SRAM, Data SRAM, Data Cache, Boot ROM, Processor-Specific MMRs

Eng. Julian S. Bruno

ADSP BF53X

Eng. Julian S. Bruno
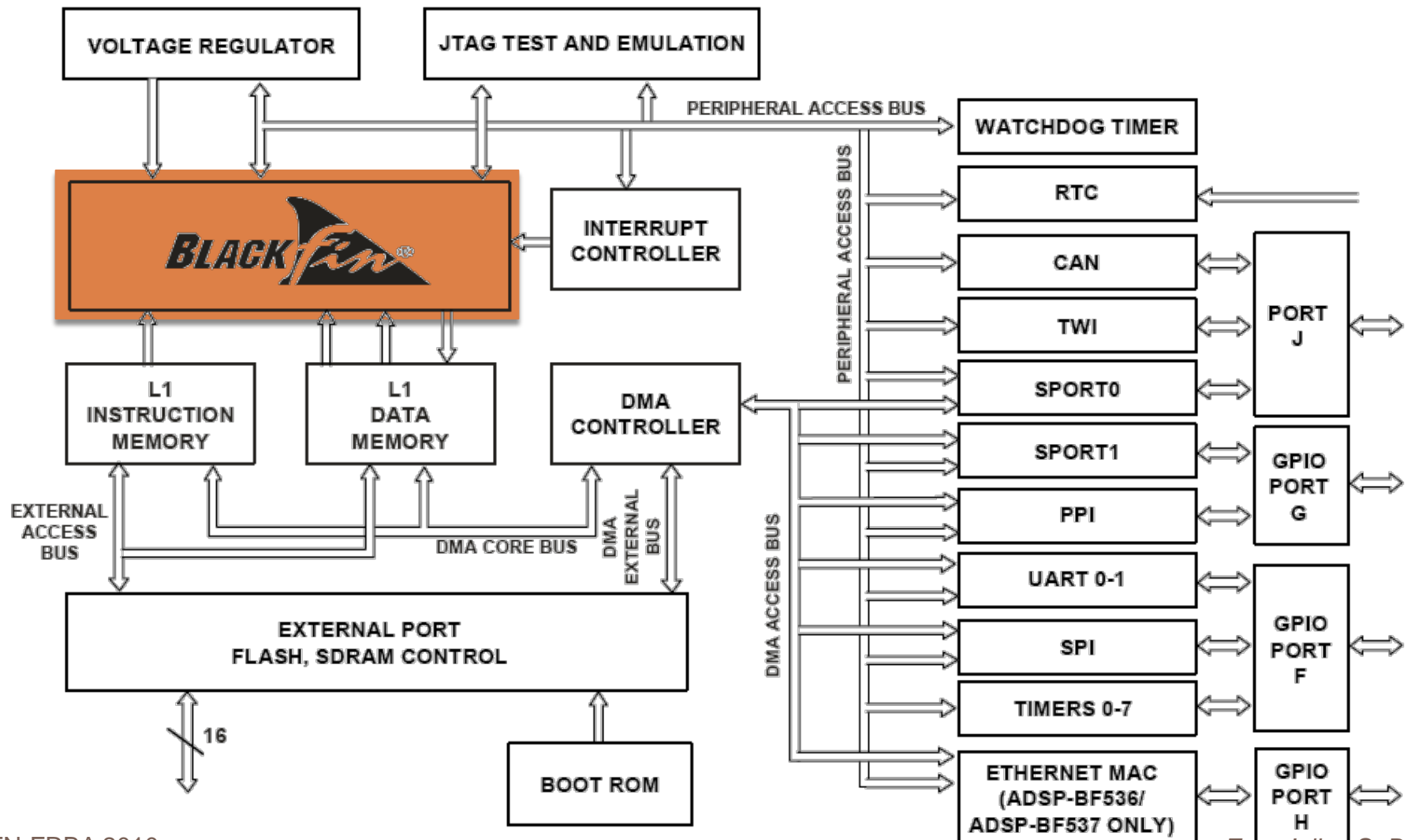
Core Architecture – BF53X

Data Arithmetic Unit

❑Two 16-bit MACs

❑Two 40-bit ALUs

❑Two 40-bit accumulators (ACC0 and ACC1)

❑Four 8-bit video ALUs

❑Single 40-bit barrel shifter

❑Data register file

❑Data types include 8-, 16-, or 32-bit signed or unsigned integer

❑Data types include 16- or 32-bit signed fractional

❑32-bit reads AND two 32-bit writes (SD, LD0, LD1)

❑Nested zero-overhead looping

❑Code density

Ad

❑Memory fetches

❑Index, length, base, and modify registers

❑Circular buffering

❑Pointer Register File, has pointers for addressing operations.

❑DAG registers

❑Stack pointer

❑Frame pointer

DATA ARITHMETIC UNIT

Eng. Julian S. Bruno

Da...53X

Six computational units:
- ❑ Two arithmetic/logic units (ALUs)
- ❑ Two multiplier/accumulator units (MACs)
- ❑ Barrel Shifter
- ❑ Set of video ALUs.

Data Register File:
- ❑ Eight registers, each 32 bits wide.
- ❑ Sixteen registers, each 16-bit wide.

Memory:
- ❑ Read two 32-bit words in each cycle (LD0-1).
- ❑ Write one 32-bit words in each cycle (SD).

Status

LOOP BUFFER

CONTROL UNIT

R7
R6
R5
R4
R3
R2
R1
R0

16   16

8   8   8   8

BARREL SHIFTER   40   40

ACC0   ACC1

CORE PROCESSOR

DATA ARITHMETIC UNIT

Eng. Julian S. Bruno

DAU

- Six computational units:
  - Two arithmetic/logic units (ALUs)
  - Two multiplier/accumulator units (MACs)
  - Barrel Shifter
  - Set of video ALUs.
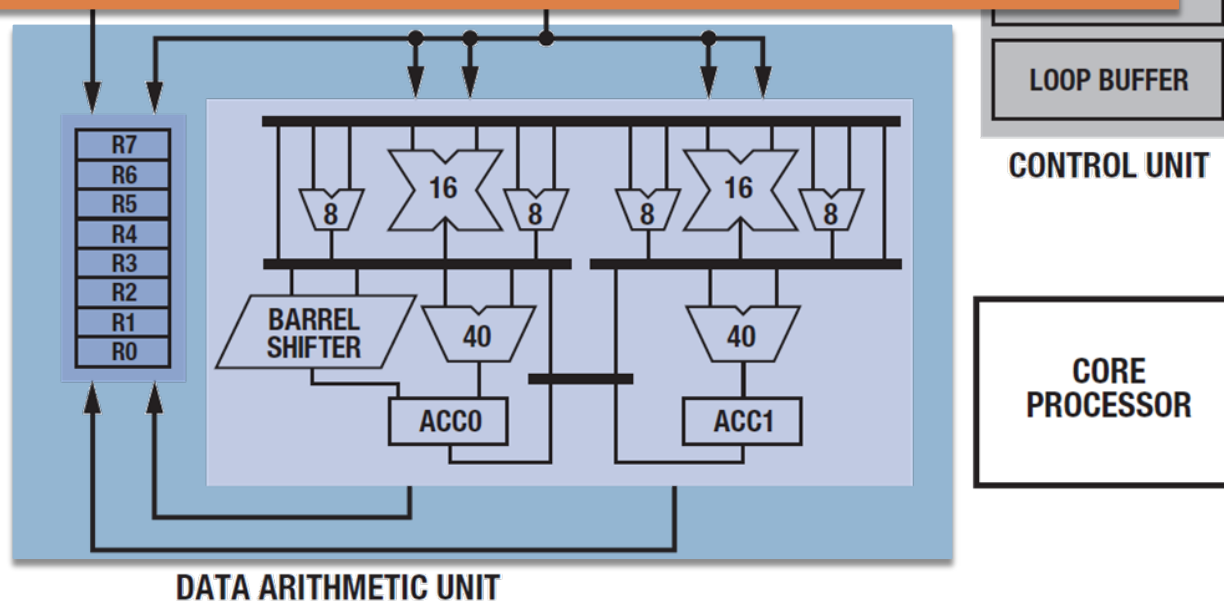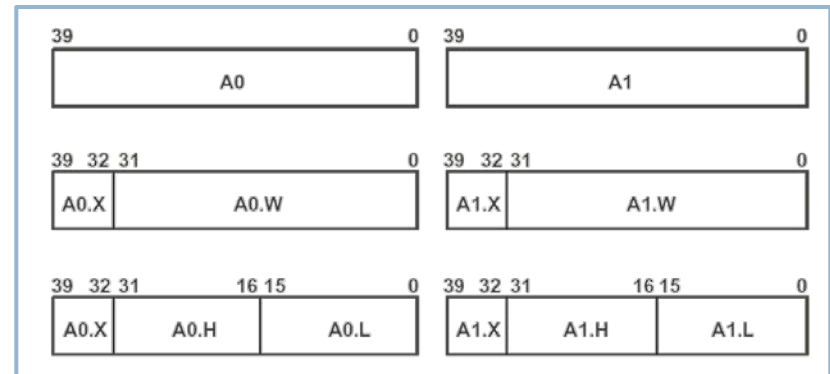- **Data Register File:**
  - **Eight registers, each 32 bits wide.**
  - **Sixteen registers, each 16-bit wide.**
- Memory:
  - Read two 32-bit words in each cycle (LD0-1).
  - Write one 32-bit words in each cycle (SD).
- Status

Eng. Julian S. Bruno

DAU - Registers

□ Register Files:
  ▫ Data Register File
    ■ R0-7 (32 bits).
    ■ Rx.H and Rx.L (16 bits).

□ Accumulator Register:
  ▫ A0 and A1(40 bits)
    ■ Ax.W (32 bits).
    ■ Ax.H and Ax.L (16 bits).
    ■ Ax.X (8 bits).

□ Both internal and external memory are accessed in little endian byte order.

Eng. Julian S. Bruno
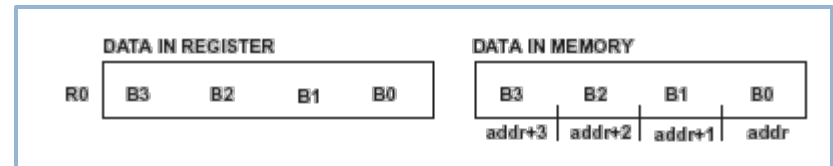
DAU - Data Formats

Format	Representation in Memory	Representation in 32-bit Register
32.0 Unsigned Word	DDDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD	DDDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD
32.0 Signed Word	SDDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD	SDDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD
16.0 Unsigned Half Word	DDDD DDDD DDDD DDDD	0000 0000 0000 0000 DDDD DDDD DDDD DDDD
16.0 Signed Half Word	SDDD DDDD DDDD DDDD	SSSS SSSS SSSS SSSS SDDD DDDD DDDD DDDD
8.0 Unsigned Byte	DDDD DDDD	0000 0000 0000 0000 0000 0000 DDDD DDDD
8.0 Signed Byte	SDDD DDDD	SSSS SSSS SSSS SSSS SSSS SSSS SDDD DDDD
1.15 Signed Fraction	S.DDD DDDD DDDD DDDD	SSSS SSSS SSSS SSSS S.DDD DDDD DDDD DDDD
1.31 Signed Fraction	S.DDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD	S.DDD DDDD DDDD DDDD DDDD DDDD DDDD DDDD
Packed 8.0 Unsigned Byte	DDDD DDDD *DDDD DDDD* DDDD DDDD *DDDD DDDD*	DDDD DDDD *DDDD DDDD* DDDD DDDD *DDDD DDDD*
Packed 1.15 Signed Fraction	S.DDD DDDD DDDD DDDD *S.DDD DDDD DDDD DDDD*	S.DDD DDDD DDDD DDDD *S.DDD DDDD DDDD DDDD*

- s = sign bit(s)
- "." = decimal point by convention
- d = data bit(s)
- Italics denotes data from a source other than adjacent bits.

Eng. Julian S. Bruno

DAU

- Six computational units:
    - Two arithmetic/logic units (ALUs)
    - Two multiplier/accumulator units (MACs)
    - Barrel Shifter
    - Set of video ALUs.
- Data Register File:
    - Eight registers, each 32 bits wide.
    - Sixteen registers, each 16-bit wide.
- Memory:
    - Read two 32-bit words in each cycle (LD0-1).
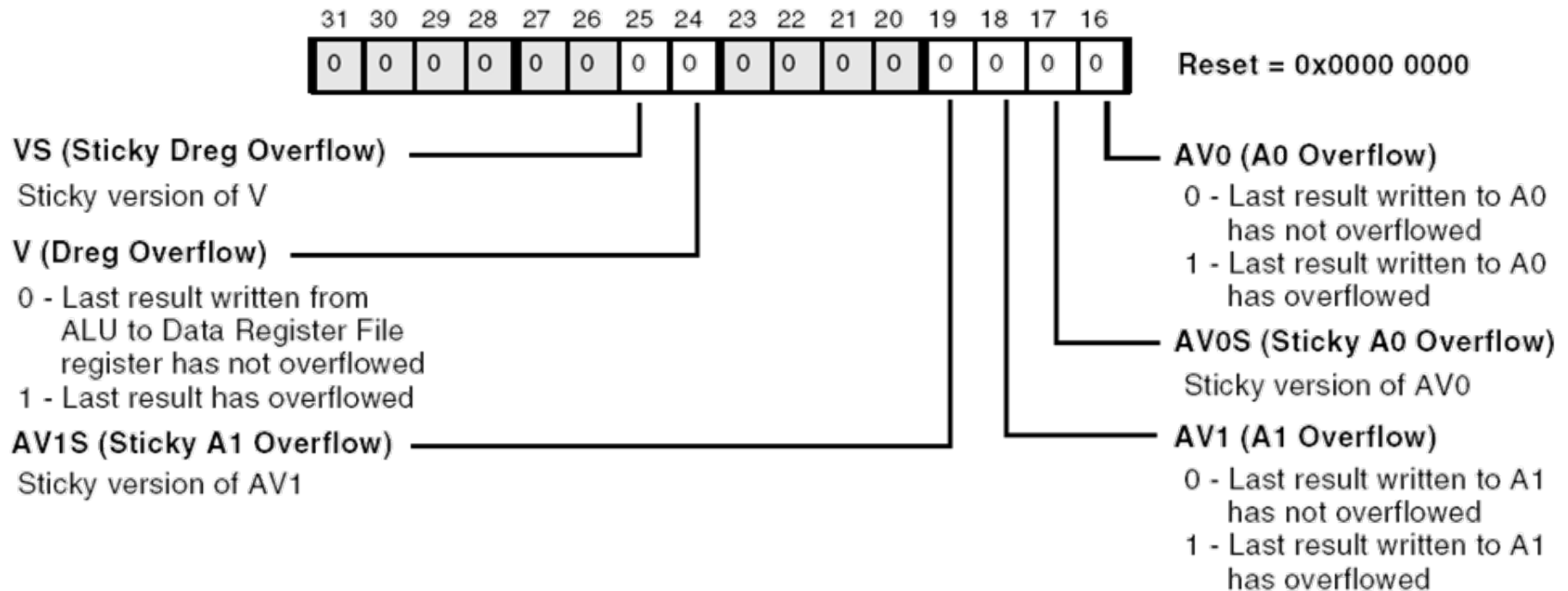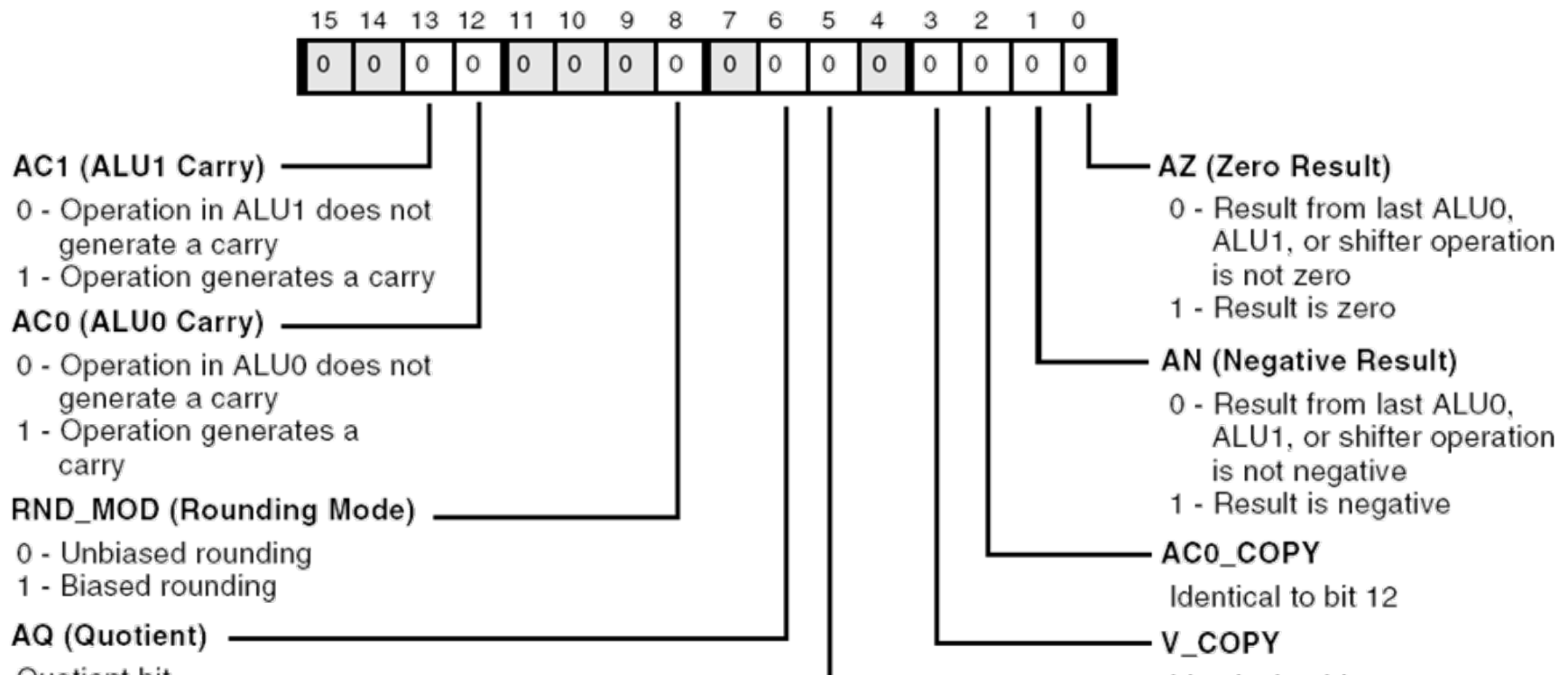   - Write one 32-bit words in each cycle (SD).
- **Status**

Eng. Julian S. Bruno

Arithmetic Status Register (ASTAT)

The logic of the overflow bits (V, VS, AV0, AV0S, AV1, AV1S) is based on two's-complement arithmetic. A bit or set of bits is set if the Most Significant Bit (MSB) changes in a manner not predicted by the signs of the operands and the nature of the operation.

Arithmetic Status Register (ASTAT)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AC1 (ALU1 Carry)
0 - Operation in ALU1 does not generate a carry
1 - Operation generates a carry

AC0 (ALU0 Carry)
0 - Operation in ALU0 does not generate a carry
1 - Operation generates a carry

RND_MOD (Rounding Mode)
0 - Unbiased rounding
1 - Biased rounding

AQ (Quotient)
Quotient bit.

AZ (Zero Result)
0 - Result from last ALU0, ALU1, or shifter operation is not zero
1 - Result is zero

AN (Negative Result)
0 - Result from last ALU0, ALU1, or shifter operation is not negative
1 - Result is negative

AC0_COPY
Identical to bit 12

V_COPY
Identical to bit 24

· **CC (Condition Code)**
Multipurpose flag, used primarily to hold resolution of arithmetic comparisons. Also used by some shifter instructions to hold rotating bits.

The logic of the carry bits (AC0, AC1) is based on unsigned magnitude arithmetic. The bit is set if a carry is generated from bit 16 (the MSB).
The carry bits (AC0, AC1) are most useful for the lower word portions of a multiword operation.

Eng. Julian S. Bruno

DAU

- **Six computational units:**
  - **Two arithmetic/logic units (ALUs)**
  - Two multiplier/accumulator units (MACs)
  - Barrel Shifter
  - Set of video ALUs.
- Data Register File:
  - Eight registers, each 32 bits wide.
  - Sixteen registers, each 16-bit wide.
- Memory:
  - Read two 32-bit words in each cycle (LD0-1).
  - Write one 32-bit words in each cycle (SD).
- Status

Eng. Julian S. Bruno

Arithmetic Logic Unit (ALU)

- ALUs perform arithmetic and logical operations on fixed-point data.

- In/out operands : 16-, 32-, and 40-bit fixed-point

- Primary ALU operations occur on ALU0, while parallel operations occur on ALU1, which performs a subset of ALU0 operations.

- ALU instructions include:

  - Fixed-point addition and subtraction of registers
  - Addition and subtraction of immediate values
  - Accumulation and subtraction of multiplier results
  - Logical AND, OR, NOT, XOR, bitwise XOR, Negate
  - Functions: ABS, MAX, MIN, Round, division primitives

Eng. Julian S. Bruno

Arithmetic Operations

- "ABS"
- "Add"
- "Add/Subtract – Prescale Down"
- "Add/Subtract – Prescale Up"
- "Add Immediate" - register += constant
- "DIVS, DIVQ (Divide Primitive)"
- "EXPADJ" - dest_reg = EXPADJ (sample_register, exponent_register)
- "MAX" - dest_reg = MAX (src_reg_0, src_reg_1)
- "MIN" - dest_reg = MIN (src_reg_0, src_reg_1)
- "Modify – Decrement" - dest_reg -= src_reg
- "Modify – Increment" - dest_reg += src_reg
- "Negate (Two's-Complement)"
- "RND (Round to Half-Word)"
- "Saturate"
- "SIGNBITS" - dest_reg = SIGNBITS sample_register
- "Subtract"
- "Subtract Immediate" - register -= constant

Eng. Julian S. Bruno

Arithmetic Operations

☐ Single 16-Bit Operations

R3.H = R1.H + R2.L (NS) ; //ALU0

☐ Dual 16-Bit Operations

R3 = R1 +|– R2 (S) ; //ALU0
// R3.H = R1.H + R2.H y R3.L = R1.L - R2.L

☐ Quad 16-Bit Operations

R3 = R0 +|+ R1, R2 = R0 –|– R1 (S) ; //ALU0 and ALU1
// The same two pairs of 16-bit inputs are presented to ALU1 as to ALU0.

☐ Single 32-Bit Operations
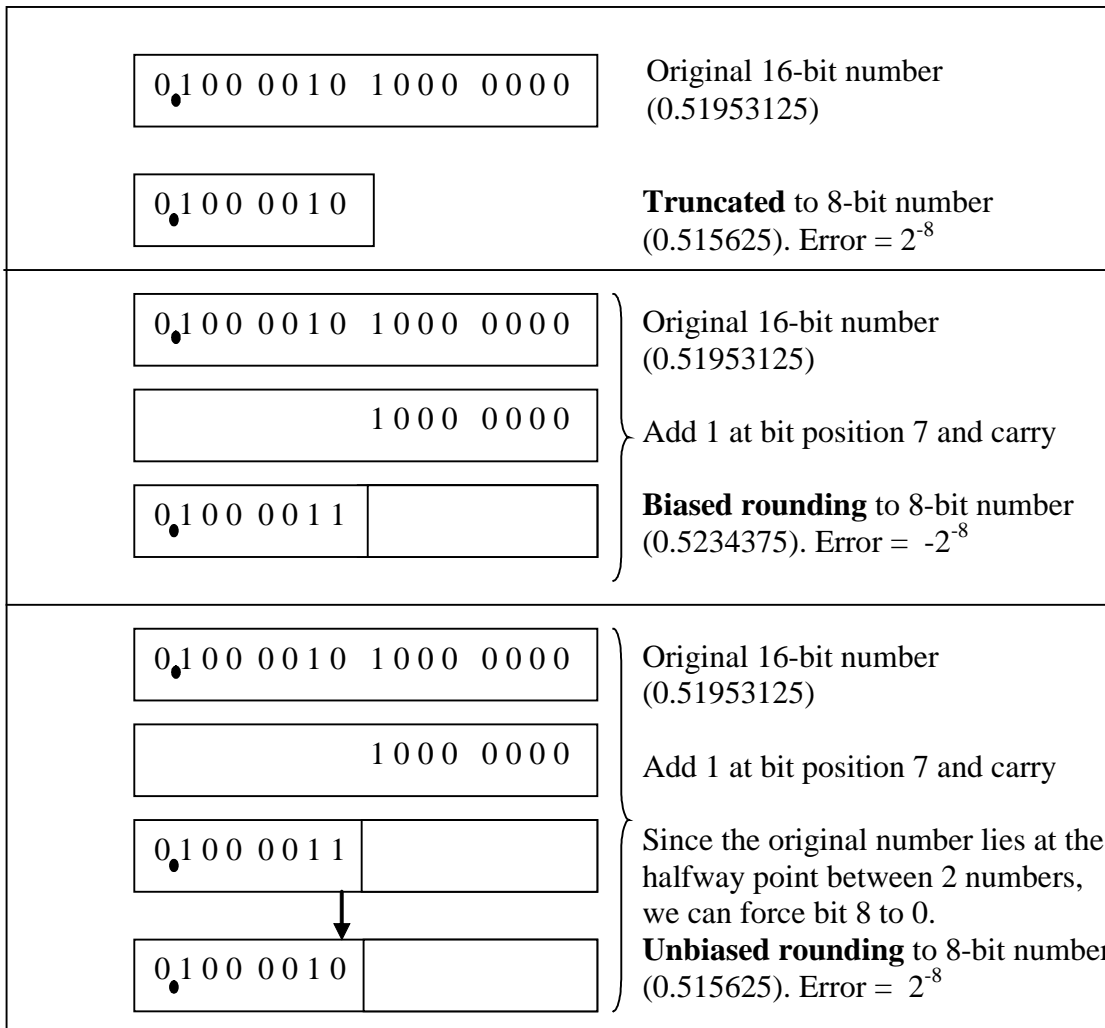
R3 = R1 + R2 (S) ; //ALU0

☐ Dual 32-Bit Operations

R3 = R1 + R2, R4 = R1 – R2 (NS) ; //ALU0 and ALU1
R3 = A0 + A1, R4 = A0 – A1 (S) ; //ALU0 and ALU1
// The same two pairs of 16-bit inputs are presented to ALU1 as to ALU0

Eng. Julian S. Bruno

Arithmetic Mode and Options for ALU

Mode	Option	Example and explanation
Dual and quad 16-bit operation: (opt_mode_0)	S	Saturate the result at 16-bit R3 = R1+|-R2 (s);
	CO	Cross option which swaps the order of the results in the destination registers for use in complex math R3 = R1+|-R2 (co);
	SCO	Combination of S and CO options
Dual 32-bit and 40-bit operation: (opt_mode_1)	S	Saturate result at 32-bit R3 = R1 + R2, R4 = R1-R2 (s);
Quad 16-bit operation: (opt_mode_2)	ASR	Arithmetic shift right which halves the result before storing to the destination register R3 = R1 +|-R2, R4 = R1-|+R2 (s,asr); Scaling is performed for the results before saturation
	ASL	Arithmetic shift left which doubles the result before storing to the destination register

Eng. Julian S. Bruno

Truncation and Rounding

0.1 0 0 0 0 1 0 1 0 0 0 0 0 0 0	Original 16-bit number (0.51953125)
0.1 0 0 0 0 1 0	**Truncated** to 8-bit number (0.515625). Error = 2^{-8}
0.1 0 0 0 0 1 0 1 0 0 0 0 0 0 0	Original 16-bit number (0.51953125)
1 0 0 0 0 0 0 0	Add 1 at bit position 7 and carry
0.1 0 0 0 0 1 1	**Biased rounding** to 8-bit number (0.5234375). Error = -2^{-8}
0.1 0 0 0 0 1 0 1 0 0 0 0 0 0 0	Original 16-bit number (0.51953125)
1 0 0 0 0 0 0 0	Add 1 at bit position 7 and carry
0.1 0 0 0 0 1 1	Since the original number lies at the halfway point between 2 numbers, we can force bit 8 to 0.
0.1 0 0 0 0 1 0	**Unbiased rounding** to 8-bit number (0.515625). Error = 2^{-8}

(T) and (TFU)

(RND)
- Bias (round-to-nearest) rounding
  RND_MOD = 1
  MATLAB: ceil()
-Unbiased (or convergent) rounding
  RND_MOD = 0
  MATLAB: round()

The RND_MOD bit of the ASTAT specifies the rounding mode.

Eng. Julian S. Bruno

Logical Operations

- "& (AND)"
- "~ (NOT One's-Complement)"
- "| (OR)"
- "^ (Exclusive-OR)"
- "BXORSHIFT, BXOR"

DAU

- **Six computational units:**
  - Two arithmetic/logic units (ALUs)
  - **Two multiplier/accumulator units (MACs)**
  - Barrel Shifter
  - Set of video ALUs.
- Data Register File:
  - Eight registers, each 32 bits wide.
  - Sixteen registers, each 16-bit wide.
- Memory:
  - Read two 32-bit words in each cycle (LD0-1).
  - Write one 32-bit words in each cycle (SD).
- Status

Eng. Julian S. Bruno

Multiplier/Accumulator (MAC)

- MAC0 and MAC1
  - Fixed-point multiplication
  - Multiply and accumulate operations are available
- Multiplier fixed-point
  - Input:16-bit fixed-point data
  - Output: 32-bit results that may be added or subtracted from a 40-bit accumulator.
  - Rounding optional
- Inputs
  - Fractional or Integer .
  - Unsigned or two's-complement.

Eng. Julian S. Bruno

Multiplier/Accumulator (MAC)

- In MAC0, both inputs are treated as signed or unsigned.

- In MAC1, there is a mixed-mode option.

- If both inputs are fractional and signed, the multiplier automatically shifts the result left one bit to remove the redundant sign bit.

- Unsigned fractional, integer, and mixed modes do not perform a shift for sign bit correction.

Eng. Julian S. Bruno

Multiplier Modes Formats

□ **Multiplier Fractional Modes Formats**

Operation	Operand Formats	Result Formats
Multiplication	1.15 explicitly signed or unsigned	2.30 shifted to 1.31
Multiplication/Addition	1.15 explicitly signed or unsigned	2.30 shifted to 1.31
Multiplication/Subtraction	1.15 explicitly signed or unsigned	2.30 shifted to 1.31

□ **Multiplier Arithmetic Integer Modes Formats**

Operation	Operand Formats	Result Formats
Multiplication	16.0 explicitly signed or unsigned	32.0 not shifted
Multiplication/Addition	16.0 explicitly signed or unsigned	32.0 not shifted
Multiplication/Subtraction	16.0 explicitly signed or unsigned	32.0 not shifted

Multiplier Instruction

□ **Multiply 16-Bit Operands**

    R3.L=R3.H*R2.H ;   /* MAC0. Both operands are signed fractions. */

    R3.H=R6.H*R4.L (FU) ;   /* MAC1. Both operands are unsigned fractions.*/

    R6=R3.H*R4.H ;   /* MAC0. Signed fraction operands, results saved as 32 bits. */

□ **Multiply 32-Bit Operands**

    R3 *= R0;

□ **Multiply and Multiply-Accumulate to Accumulator**

    A0=R3.H*R2.H ;   /* MAC0, only. Both operands are signed fractions.*/

    A1+=R6.H*R4.L (FU) ;   /* MAC1, only. Both operands are unsigned fractions. */

□ **Multiply and Multiply-Accumulate to Half-Register**

    R3.L=(A0=R3.H*R2.H) ;   /* MAC0, only. Both operands are signed fractions. */

    R3.H=(A1+=R6.H*R4.L) (FU) ; /* MAC1, only. Both operands are unsigned fractions. */

□ **Multiply and Multiply-Accumulate to Data Register**

    R4=(A0=R3.H*R2.H) ;   /* MAC0, only. Both operands are signed fractions. */

    R3=(A1+=R6.H*R4.L) (FU) ;   /* MAC1, only. Both operands are unsigned fractions.*/

□ **Dual MAC Operations**

    A1 += R1.H * R2.L, A0 += R1.L * R2.H;

    R3.H = (A1 += R1.H * R2.L), R3.L = (A0 += R1.L * R2.L);

Eng. Julian S. Bruno

Multiplier Instruction Options

- (FU) Input data operands are unsigned fraction. No shift correction is made.

- (IS)   Input data operands are signed integer. No shiftcorrection is made.

- (IU) Input data operands are unsigned integer. No shift correction is made.

- (T) Input data operands are signed fraction. When copying to the destination half register, truncates the lower 16 bits of the Accumulator contents.

- (TFU) Input data operands are unsigned fraction. When copying to the destination half register, truncates the lower 16 bits of the Accumulator contents.

- (ISS2) the number is saturated to its maximum positive or negative value.

- (IH) This option indicates integer multiplication with high half word extraction.

- (W32) Input data operands are signed fraction with no extension bits in the Accumulators at 32 bits.

- (M) Operation uses mixed-multiply mode. Valid only for MAC1 versions of the instruction.

Eng. Julian S. Bruno

Multiplier Data Flow Details

Eng. Julian S. Bruno
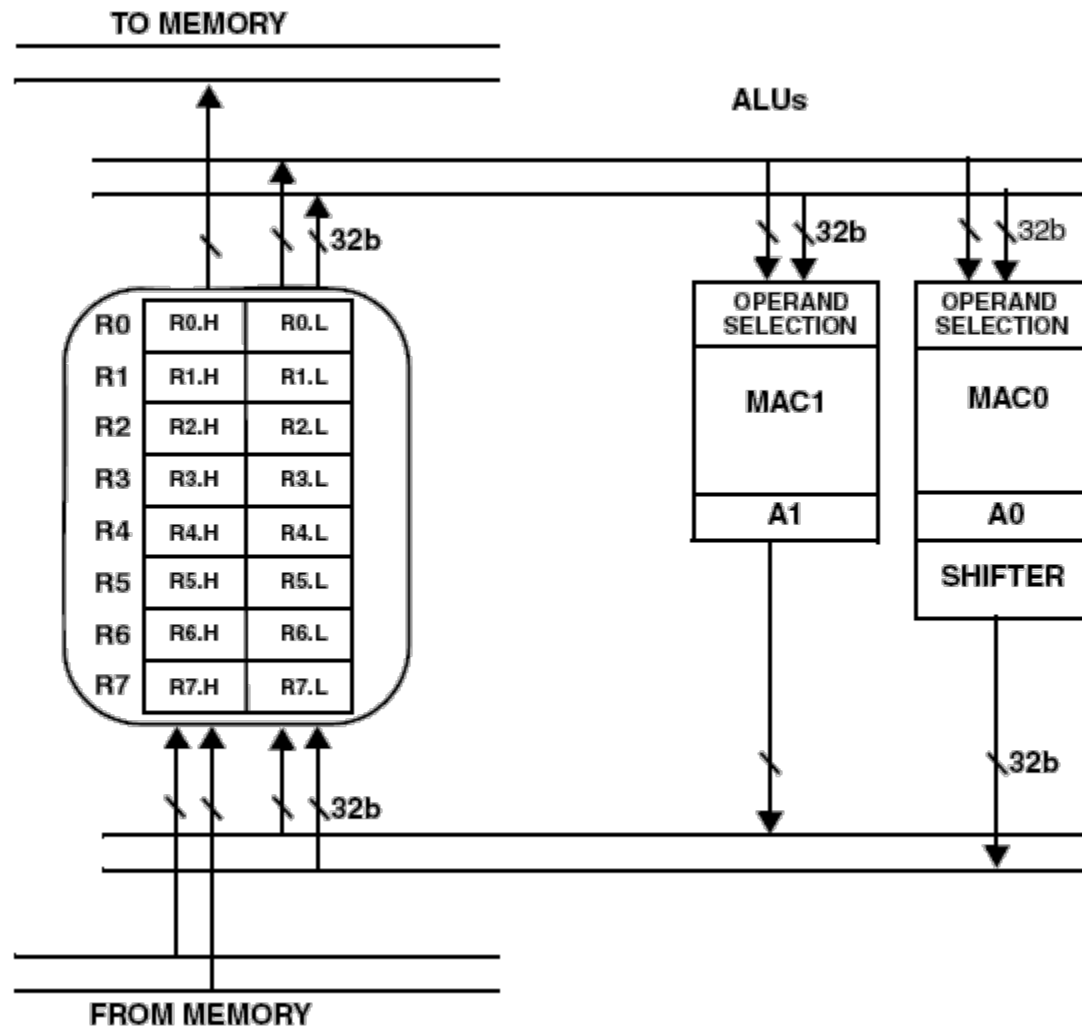
MAC0 combined with MAC1

- Both scalar instructions must share the same mode option (for example, default, IS, IU, T).

- Both scalar instructions must share the same pair of source registers, but can reference different halves of those registers.

- If both scalar operations write to destination D-registers, they must write to the same sized destination D-registers, either 16 or 32 bits.

- The destination D-registers (if applicable) for both scalar operations must form a vector couplet, as described below:

  - 16-bit: store the results in the upper- and lower-halves of the same 32-bit Dreg. MAC0 writes to the lower half, and MAC1 writes to the upper half.

    - *R3.H = (A1 += R1.H * R2.L), R3.L = (A0 += R1.L * R2.L);*

  - 32-bit: store the results in valid Dreg pairs. MAC0 writes to the pair's lower (even-numbered) Dreg, and MAC1 writes to the upper (odd-numbered) Dreg.

    - *R5 = (A1 += R1.H * R2.L) , R4 = (A0 += R1.L * R2.L) (IS);*

DAU

- **Six computational units:**
  - Two arithmetic/logic units (ALUs)
  - Two multiplier/accumulator units (MACs)
  - **Barrel Shifter**
  - Set of video ALUs.
- Data Register File:
  - Eight registers, each 32 bits wide.
  - Sixteen registers, each 16-bit wide.
- Memory:
  - Read two 32-bit words in each cycle (LD0-1).
  - Write one 32-bit words in each cycle (SD).
- Status

Eng. Julian S. Bruno

Barrel Shifter

- Functions
  - arithmetic shift
  - logical shift
  - rotate
  - bit test
  - set
  - pack
  - unpack
  - exponent detection
- Inputs: 16-, 32-, or 40-bit
- Outputs: 16-, 32-, or 40-bit

Shift/Rotate Operations

- "Add with Shift" combines an addition operation with a one- or two-place logical shift left

- "Shift with Add" combines a one- or two-place logical shift left with an addition operation. Useful for array pointer manipulation

- "Arithmetic Shift" Ashift, >>>, >>>=, <<(s), opt_sat

- "Logical Shift" Lshift, >>, >>=, <<, =<<

- "ROT (Rotate)" rotates a register through the CC bit

- Two-Operand or Three-Operand Shifts
- Immediate or Register Shifts

Eng. Julian S. Bruno

Two-Operand Shifts

□ Immediate Shifts

// R0 contains 0000 B6A3 ;

R0 >>= 0x04 ;

// R0 contains 0000 0B6A ;

□ Register Shifts

// R0 contains 0000 B6A3 and R2 contains 0000 0004 ;

R0 <<= R2 ;

// R0 contains 000B 6A30 ;

Three-Operand Shifts

☐ ## Immediate Shifts

// R0.L contains B6A3 ;

R1.H = R0.L << 0x04 ;

// R1.H contains 6A30 ;

☐ ## Register Shifts

// R0 contains 0000 B6A3 and R2.L contains 0004

R1 = R0 ASHIFT by R2.L ;

// R1 contains 000B 6A30 ;

// R0 contains ABCD EF12 , R2.L contains 0004 and CC=0

R1 = R0 ROT by R2.L ;

// R1 contains BCDE F125 ;

Eng. Julian S. Bruno

Bit Operations

- "BITCLR"
- "BITSET"
- "BITTGL"
- "BITTST"
- "DEPOSIT" merges the background bit field with the foreground bit field.
- "EXTRACT" moves only specific bits from the scene_reg into the low-order bits of the dest_reg
- "BITMUX" merges bit streams
- "ONES (One's-Population Count)"

Vector Operations

- "Add on Sign"
- "VIT_MAX (Compare-Select)"
- "Vector ABS"
- "Vector Add / Subtract"
- "Vector Arithmetic Shift"
- "Vector Logical Shift"
- "Vector MAX"
- "Vector MIN"
- "Vector Multiply"
- "Vector Multiply and Multiply-Accumulate"
- "Vector Negate (Two's-Complement)"
- "Vector PACK"
- "Vector SEARCH"

Eng. Julian S. Bruno

Control Code Bit Management

□ **"Compare Data Register" or "Compare Pointer"**

CC = operand_1 == operand_2

CC = operand_1 < operand_2

CC = operand_1 <= operand_2

CC = operand_1 < operand_2 (IU)

CC = operand_1 <= operand_2 (IU)

□ **"Compare Accumulator"**
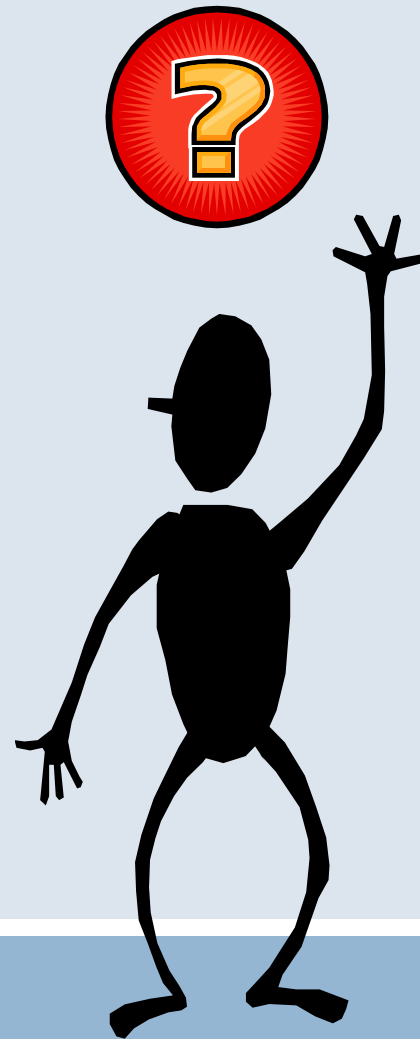
CC = A0 == A1

CC = A0 < A1

CC = A0 <= A1

□ **"Move CC"**

□ **"Negate CC"**

Also see: IF CC JUMP and IF !CC JUMP

Eng. Julian S. Bruno

Recommended bibliography

- Blackfin Processor Programming Reference, Revision 1.3, September 2008
  - Ch2: Computational Units
  - Ch11: CONTROL CODE BIT MANAGEMENT
  - Ch12: LOGICAL OPERATIONS
  - Ch13: BIT OPERATIONS
  - Ch14: SHIFT/ROTATE OPERATIONS
  - Ch15: ARITHMETIC OPERATIONS

- NOTE: Many images used in this presentation were extracted from the recommended bibliography.

Eng. Julian S. Bruno

Questions?

Thank you!