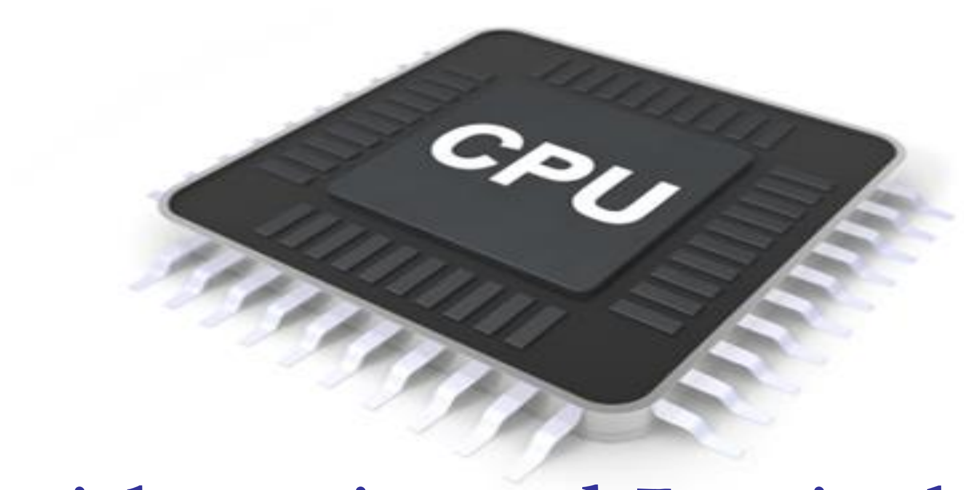

Arithmetic and Logical Ops

ENEE 3582

Microp

Byte Addition: ADD ADC

- ADD
  - Add 2 registers, store the result in the destination reg
  - Rd, Rs can be `R0…R31`
  - Syntax: `ADD Rd, Rs ;Rd = Rd + Rs`
- ADC
  - Add 2 registers and the carry flag (CF), store the result in the destination reg
  - Rd, Rs can be `R0…R31`
  - CF is in SREG.
    - based on the carry from the last operation
  - Syntax: `ADC Rd, Rs ;Rd = Rd + Rs + CF`

Byte Subtraction: SUB SBC

- SUB
  - Subtracts sources register (Rs) **from** destination reg (Rd)
  - Rd, Rs can be `R0…R31`
  - Syntax: `SUB Rd, Rs ;Rd = Rd - Rs`
- SBC
  - Subtracts sources register and CF **from** destination reg
  - Rd, Rs can be `R0…R31`
  - Syntax: `SBC Rd, Rs ;Rd = Rd - Rs - CF`

Byte Increment/Decrement

- `INC`
  - Increments the dest reg by 1
  - Rd, Rs can be `R0…R31`
  - Syntax: `INC Rd ;Rd = Rd + 1`
- `DEC`
  - Rd, Rs can be `R0…R31`
  - Syntax: `DEC Rd ;Rd = Rd - 1`

Byte Subtraction with Immediate: SUBI SBCI DEC

- SUBI
  - Rd can only be R16…R31
  - K is 8bit (0 to 255)
  - Syntax: `SUBI Rd, K` ;Rd = Rd – K
- SBCI
  - Rd can only be R16…R31
  - Syntax: `SBCI Rd, K` ;Rd = Rd - K - CF
- For ADD with immediate:
  - Implement as `SUBI Rd, -K` ;Rd = Rd –(-K) = Rd + K

Word Addition, Subtraction: ADIW SBIW

- Useful for X, Y, Z operations
  - X = R27:R26
  - Y = R29:R28
  - Z = R31:R30
- Add/Sub an immediate (K) to/from word stored in 2 registers
- K is 6 bits (0 to 63)
- Rd can **<u>only</u>** be R24, R26, R28, R30
- Results is stored in the 2 registers
- Syntax: `ADIW Rd+1:Rd, K ;Rd+1:Rd = Rd+1:Rd + K`
- Syntax: `SBIW Rd+1:Rd, K ;Rd+1:Rd = Rd+1:Rd - K`

Integer Multiplication

- Unsigned 8-bit multiplication
  - 0x00 * 0xFF = 0 * 255 = 0 = 0x0000
  - 0x7F * 0xFF = 127 * 255 = 32385 = 0x7E81
  - 0x80 * 0xFF = 128 * 255 = 32640 = 0x7F80
  - 0xFF * 0xFF = 255 * 255 = 65025 = 0xFE01
- Signed 8-bit multiplication
  - 0x00 * 0xFF = 0 * -1 = 0 = 0x0000
  - 0x7F * 0xFF = 127 * -1 = -127 = 0xFF81
  - 0x80 * 0xFF = -128 * -1 = +128 = 0x0080
  - 0xFF * 0xFF = -1 * -1 = 1 = 0x0001
- For n bit multiplication we need 2n results
  - 8-bit multiplication needs 16-results

Multiplication: MUL MULS MULSU

- MUL
  - Unsigned multiplication
  - Syntax: `MUL Ru,Rv ;R1:R0 = Ru*Rv.`
- MULS
  - Signed multiplication
  - Syntax: `MULS Rs,Rt ;R1:R0 = Rs*Rt`
- MULSU
  - Signed with unsigned multiplication
  - Syntax: `MULSU Rs,Ru ;R1:R0 = Rs*Ru`
  - Answer will be signed
  - Can use the unsigned range for multiplication
- Ru,Rv,Rs,Rt are registers general purpose regs R0…R31

Examples

```
LDI R16, 0xFF
LDI R17, 0x80
LDI R18, 0x7F

MUL R16,R16
MUL R16,R17
MUL R16,R18

MULS R16,R16
MULS R16,R17
MULS R16,R18

MULSU R16,R16       ;-1*255 = -255 = 0xFF01
MULSU R16,R17       ;-1*128 = -128 = 0xFF80
MULSU R16,R18       ;-1*127 = -127 = 0xFF81
```

Real Binary

- Given any unsigned real number in base x:

$$(a_{n-1}a_{n-2}a_{n-3}\ldots a_2a_1a_0).(a_{-1}a_{-2}a_{-3}\ldots a_{-(m-2)}a_{-(m-1)}a_{-m})$$

  a_i = 0, 1, …., x-1.

  n numbers before the decimal, m numbers after decimal.

  Binary: x = 2 ; a_i = 0 or 1.

- To convert to base 10:

$$(a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots\ a_1x^1 + a_0x^0) + (a_{-1}x^{-1} + a_{-2}x^{-2} + \ldots\ a_{-(m-1)}x^{-(m-2)} + a_{-m}x^{-m})$$

- Example: 1011.1011 b

m :					1	2	3	4
n :	4	3	2	1				
a_i :	a_3	a_2	a_1	a_0	a_{-1}	a_{-2}	a_{-3}	a_{-4}
	1	0	1	1 .	1	0	1	1
	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}

Converting from Binary to Decimal

2^n	Binary	Fraction	Decimal
2^{-1}	0b0.1	1/2	0.5
2^{-2}	0b0.01	1/4	0.25
2^{-3}	0b0.001	1/8	0.125
2^{-4}	0b0.0001	1/16	0.0625
2^{-5}	0b0.00001	1/32	0.03125
2^{-6}	0b0.000001	1/64	0.015625
2^{-7}	0b0.0000001	1/128	0.0078125

1011.1011 b = $1*2^3+0*2^2+1*2^1+1*2^0+1*2^{-1}+0*2^{-2}+1*2^{-3}+1*2^{-4}$

= 8 +2 +1 +1/2 +1/8 +1/16

= 11.6875d or 11 11/16

Converting to decimal is easy. Example:

1101.100101b → 1101b = 13d, 0.100101b = 100101b/2^6d = 37/64 = 13 37/64

Real to Decimal: Quick Conversion

- Steps:
  - Ignore decimal
  - Convert binary to decimal
  - Divide by 2^n to move decimal point back n places
- Examples:
  - 0b100.1001 = $\frac{0b1001001}{2^4} = \frac{73}{16} = 4.5625$
  - 0b11110.1101 = $\frac{0b111101101}{2^4} = \frac{493}{16} = 30.8125$
  - 0b111.101101 = $\frac{0b111101101}{2^6} = \frac{493}{64} = 7.703125$

1.7 Fractional Binary Format

- N.Q Format
  - N: integer bits
  - Q: Fraction bits
- 1.7 Format:
  - 1 integer bit (MSB)
  - 7 fraction bits
  - Simply convert number to decimal and divide by 128
  - E.g.: $\mathtt{0b00101111} \xrightarrow[1.7]{} \mathtt{0b0.0101111} = \frac{0b00101111}{2^7} = \frac{47}{128} = 0.3671875$
- Not all decimal fraction can be represented.
  - Only powers of 0.5

1.7 Fractional Unsigned Binary

- Convert number to decimal and divide by 128
- $\texttt{0b00000000} = \frac{0b00000000}{128} = 0.0$
- $\texttt{0b10000000} = \frac{0b10000000}{128} = \frac{128}{128} = 1.0$
- $\texttt{0b01111111} = \frac{0b01111111}{128} = \frac{127}{128} = 0.9921875$
- $\texttt{0b00000001} = \frac{0b00000001}{128} = \frac{1}{128} = 0.0078125$
- $\texttt{0b11111111} = \frac{0b11111111}{128} = \frac{255}{128} = 1.9921875$

- Min, Max: 0.0, 1.9921875

Converting from Decimal Fraction to Binary

- Use successive multiplication to convert fractions to binary:
  - multiple by base (2).
  - Record and remove integer.
  - Stop when the fraction is zero.
  - Do **not** reverse order.
- Example: 0.625d = 0.101b

  0.625x2 = (1).25

  0.25x2 = (0).50

  0.5x2 = (1).0
- Many decimal fractions are infinite in binary
  - Use rounding

1.7 Fractional Signed Binary

- Binary number is signed binary (2s complement)
- Convert number to signed decimal and divide by 128
- 0b00000000 = $\frac{0b00000000}{128}$ = 0.0
- 0b10000000 = $\frac{0b10000000}{128} = \frac{-128}{128}$ = -1.0
- 0b01111111 = $\frac{0b01111111}{128} = \frac{127}{128}$ = 0.9921875
- 0b00000001 = $\frac{0b00000001}{128} = \frac{1}{128}$ = 0.0078125
- 0b11111111 = $\frac{0b11111111}{128} = \frac{-1}{128}$ = -0.0078125
- 0b11111111 = $\frac{0b10101010}{128} = \frac{-86}{128}$ = -0.671875
- Min, Max: -1, 0.9921875

Converting to Signed 1.7 Fraction

- To convert to signed 1.7 fraction format
  - Multiply the signed fraction by 128
  - Ignore all digits after the decimal
  - Convert the integer part to signed binary
- Examples:
  - -0.625: -0.625x128 = -40 = 0b11011000
  - 0.525: 0.525x128 = 67.2 ~= 67 = 0b01000011
  - -1.5: -1.5x128 = number outside range (min value is -1)
  - -0.123: -0.123x128 = -15.744 ~= -15 = 0b11111001

Functions INT() FRACT() Q7() Q15()

- Microchip Studio built-in functions
- INT(expression) = decimal integer of expression
- FRAC(expression) = decimal fraction of expression
- Q7(expression) = 1.7 Format of expression
- Q15(expression) = 1.15 Format of expression
- Examples:

```
            LDI R16, INT(-33.5)        ;INT(-33.5)=-33
            LDI R17, Q7(FRAC(-1.5))    ;-0.5 = 1.100 0000 =>R17=0xC0
FRACT15:    .DW   Q15(-0.65625)
```

1.15 Fraction Format

- AVR: 1.7 Fraction multiplication yields a 1.15 format
- MSB is integer, 15 bits fraction
- To convert to decimal: divide by 2^{15} (32768)

1.15 Format	Unsigned Value	Signed Value
`0b0000000000000000`	$\frac{0}{32768}$ = 0.0 (min)	$\frac{0}{32768}$ = 0.0
`0b1000000000000000`	$\frac{32768}{32768}$ = 1.0	$\frac{-32768}{32768}$ = -1.0 (min)
`0b0111111111111111`	$\frac{32767}{32768}$ = 0.999969482421875	$\frac{32767}{32768}$ = 0.999969482421875 (max)
`0b0000000000000001`	$\frac{1}{32768}$ = 0.000030517578125	$\frac{1}{32768}$ = 0.000030517578125
`0b1111111111111111`	$\frac{65535}{32768}$ = 1.999969482421875 (max)	$\frac{-1}{32768}$ = -0.000030517578125
`0b1010101010101010`	$\frac{43690}{32768}$ = 1.33331298828125	$\frac{-21846}{32768}$ = -0.66668701171875

Fractional Multiplication: FMUL FMULS FMULSU

- 1.7 Fractional Format multiplication
  - 1.15 Fractional Format results
  - 2 Source regs: `Rm,Rn` can be `R16…R23`
  - Result in `R1:R0`
- FMUL: Fractional multiply unsigned with unsigned
  - Syntax: `FMUL Rm, Rn ;R1:R0 = Rm x Rn`
- FMULS: Fractional multiply signed with signed
  - Syntax: `FMULS Rm, Rn ;R1:R0 = Rm x Rn`
- FMULSU: Fractional multiply signed with unsigned
  - Syntax: `FMULSU Rm, Rn ;R1:R0 = Rm x Rn`

Application 1: Multiplying Fractions

- Fractional multiplication can be used to multiply fractions
  - Range for fractions and their answer:
    - 0 to <2 (unsigned)
    - -1 to <1 (signed)
- Example:
  - Multiply 0.25 by 0.625:

```
LDI R16, Q7(0.25)
LDI R17, Q7(0.625)
FMUL R16, R17
```

  - Multiply -0.25 by -0.625:

```
LDI R16, Q7(-0.25)
LDI R17, Q7(-0.625)
FMULS R16, R17
```

Multiply -0.25 by 1.625:

```
LDI R16, Q7(-0.25)
LDI R17, 0x80+Q7(0.625)    ;0x80=10000000
FMULSU R16, R17
```

Multiply 1.05 by 1.62:

```
LDI R16, 0x80+Q7(0.05)
LDI R17, 0x80+Q7(0.62)
FMUL R16, R17
```

Application 2: Integer Division

- Integer division as fractional multiplication
  - R01 = integer
  - R00 = fraction
  - Range of X, Y ?

$$\frac{X}{Y} = X \times \frac{1}{Y} = \mathrm{R01{:}R00}$$

- Examples:
  - Divide 205 by 23

```
LDI R16, 205
LDI R17, Q7(1.0/23)
FMUL R16, R17
```

  - Divide -64 by -32

```
LDI R16, -64
LDI R17, Q7(-1.0/32)
FMULS R16, R17
```

Divide -105 by 23:

```
LDI R16, -105
LDI R17, Q7(1.0/23)
FMULSU R16, R17
```

Divide 127 by -32:

```
LDI R16, 128
LDI R17, Q7(-1.0/32)
FMULSU R17, R16
```