

SPRING 2016 CMPE 364

Microprocessor Based Design

Dr. Ryan Riley

(Slides adapted from Dr. Mohamed Al-Meer)

MULTIPLICATION

- ARM has hardware support for different Multiply instructions
 - signed, unsigned, 32-bit, 64-bit, and Multiply And Accumulate.
- multiply the contents of a pair of registers and, depending upon the instruction, accumulate the results in with another register.
- The long multiplies accumulate onto a pair of registers representing a 64-bit value.
- The final result is placed in a destination register or a pair of registers.

MULTIPLICATION

Syntax: `MLA{<cond>}{S} Rd, Rm, Rs, Rn`
`MUL{<cond>}{S} Rd, Rm, Rs`

MLA	multiply and accumulate	$Rd = (Rm * Rs) + Rn$
MUL	multiply	$Rd = Rm * Rs$

Syntax: `<instruction>{<cond>}{S} RdLo, RdHi, Rm, Rs`

SMLAL	signed multiply accumulate long	$[RdHi, RdLo] = [RdHi, RdLo] + (Rm * Rs)$
SMULL	signed multiply long	$[RdHi, RdLo] = Rm * Rs$
UMLAL	unsigned multiply accumulate long	$[RdHi, RdLo] = [RdHi, RdLo] + (Rm * Rs)$
UMULL	unsigned multiply long	$[RdHi, RdLo] = Rm * Rs$

MUL Instruction

- **MUL** multiply two 32-bit source operands, **rs** and **rm** and stores 32-bit result in destination register **rd**.
 - source operand could be signed or unsigned.
 - The 3 operands must be registers.
 - rs may NOT be shifted or rotated.
 - rd and rm must NOT be the same.
 - rd and rs could be the same
- Syntax
MUL rd, rm, rs

MUL Instruction

- EXAMPLE
 - **MUL R5, R2, R9**
 - R2 = 0X0005A014, R9= 0x00000174
- SOLUTION
 - R5 = 0x082C9D10
- EXAMPLE
 - **MUL R0, R2, R0**
 - R0 = 0xFFFFFFFFE, R2 = 0xFFFFFFFF
- SOLUTION
 - R0 = 00000002
- **MUL is a signed multiplication**

UMULL Instruction

- **Unsigned multiply long** instruction
 - multiplies 2 unsigned 32-bit source registers **rs**, **rm**
 - 64-bit result is saved in 2 32-bit destination registers **rdhi**, **rdlo**
 - rdlo, rdhi and first operand rm must be different
 - both source operands must be registers
 - rs cannot be shifted or rotated
- Syntax

UMULL rdlo, rdhi, rm, rs

UMULL Instruction

- EXAMPLE
 - **UMULL R5, R4, R8, R0**
 - R8 = 0x08065310, R0 = 0x00410EEF
- SOLUTION
 - **[R5, R4] = R8 * R0**
 - Result **0X00020A12ED826BF0**
 - R4 = 0x00020A12, R5 = 0xED826BF0

SMULL Instruction

- **Sign multiply long** instruction
 - multiplies 2 signed 32-bit numbers in source registers **rm, rs** and places 64-bit result in a pair destination registers **rdhi, rdlo**
 - source operands rs and rm must be registers
 - rs cannot be shifted or rotated
- Syntax

SMULL rdlo, rdhi, rm, rs

SMULL Instruction

- EXAMPLE
 - **SMULL R10, R9, R2, R4**
 - R2 = FFFFFFFF4F, R4 = 000000A0
- SOLUTION
 - **[R10, R9] = R2 * R4**
 - R2 = -177, R4 = 160, RES = -177 * 160 = -28,320
 - RES = FFFFFFFF FFFF9160
 - R9 = FFFFFFFF, R10 = FFFFFFFF 9160

MLA Instruction

- **Multiply and Accumulate** instruction
 - multiplies the first 2 source operands **rm**, **rs** and adds the third source operand **rn** to the **product**
 - 32-bit result is stored in rd
 - **rd = rm * rs + rn**
 - rm and rd must be different
- Syntax

MLA rd, rm, rs, rn

MLA Instruction

- EXAMPLE
 - **MLA R0, R4, R3, R6**
 - R3 = 00000018, R4 = 0000C801, R6 = 00000005
- SOLUTION
 - $R0 = 00000018 * 0000C801 + 00000005$
 - R0 = 0012C01D

UMLAL Instruction

- **Unsigned MuLtiplies Accumulate Long** instruction
 - multiplies unsigned 32-bit values in rm,rs to obtain 64-bit product
 - product is added to the 64-bit value stored in rdhi, rdlo
 - **$[Rdlo, Rdhi] = [Rdlo, Rdhi] + rm * rs$**
 - no immediate data at all
 - rs cannot be shifted or rotated
 - rdlo rdhi, rm cannot be the same
- Syntax

UMLAL rdlo, rdhi, rm, rs

UMLAL Instruction

- EXAMPLE
 - **UMLAL R2, R3 R1, R8**
 - R1= 10080040, R2 = 000A6C20, R3 = 00008000, R8 = 00000560
- Solution
 - $R1 \times R8 = 000000562B015800$
 - $[R3 _ R2] = 000000562B015800 + 00008000000A6C20$
 - $\quad\quad\quad = 000080562B0BC420.$

SMLAL Instruction

- **Signed Multiply Accumulate Long** Instruction
 - multiplies 2 signed 32-bit numbers in **rm** and **rs** and
 - 64-bit product is added to 64-bit value stored in register pair **rdlo** and **rdhi**.
 - $[Rdlo, Rdhi] = [Rdlo, Rdhi] + rm*rs$
 - all operands are registers
 - rs cannot be shifted or rotated
 - rdlo, rdhi, and rm must be different.
- Syntax

SMLAL rdlo, rdhi, rm, rs

SMLAL Instruction

- EXAMPLE
 - **SMLAL R4, R5, R1, R6**
 - R1 = 680EC921, R4 = FFFFFFFF, R5 = FFFFFFFF,
 - R6 = FFF80041
- SOLUTION
 - $[R4, R5] = [R4, R5] + R1 * R6$.
 - $[R4, R5] = \text{FFFCBFA4 22B91107}$

DIVISION

- No Division instruction in ARM.
- must be carried out using successive subtraction
- see book Figure in page41 [Schindler]