Chapter 15

ARM –
Architecture, Programming
and
Development Tools

Lesson 5

ARM 16-bit Thumb Instruction Set

- **Thumb**®16 bit subset

- Better code density than 32-bit architecture instruction set

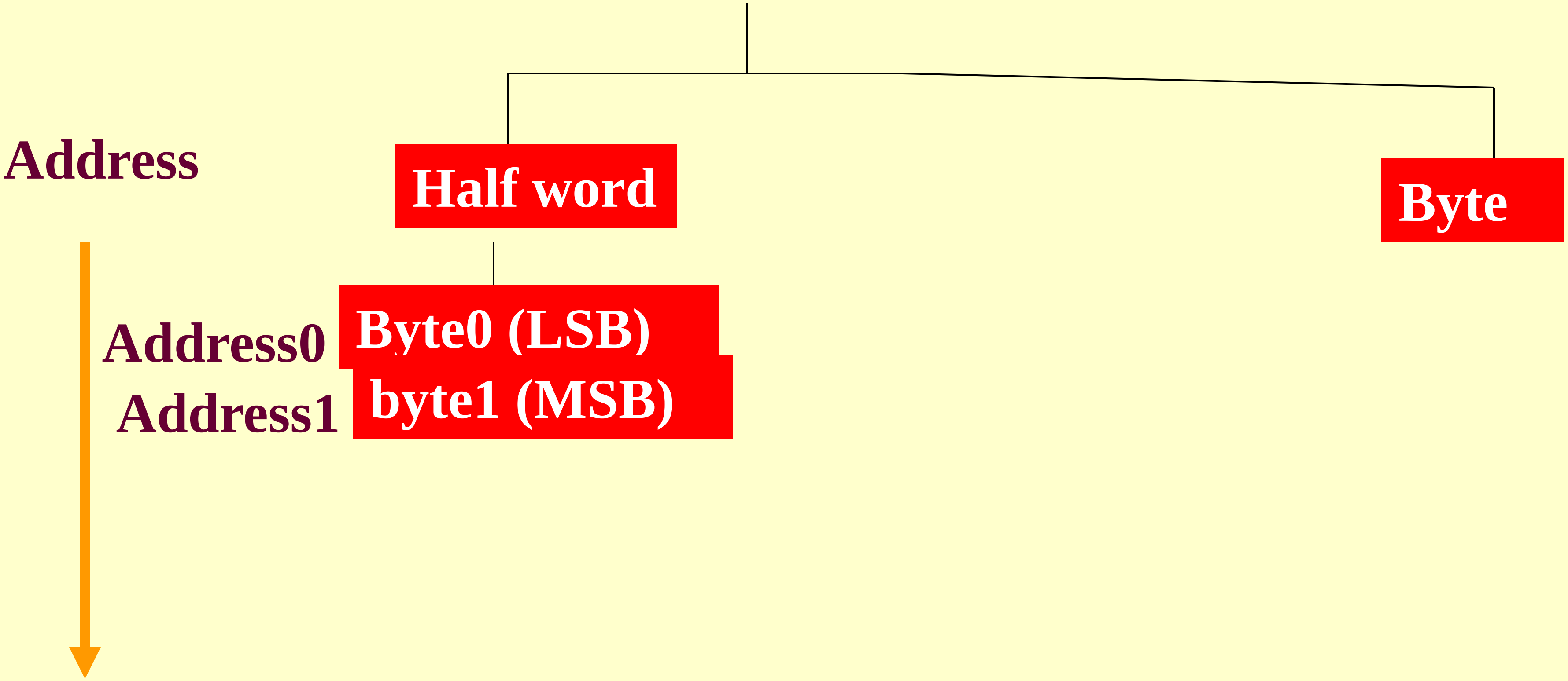
Basic Programming Features-

- Processor can operate on the words as per initialization.
- A half word alignment can in big endian [least significant byte stored as higher bits (address 1) of a word] or little endian [least significant byte stored as lower bits (address 0) of a word].

Data Types support

- 8-bit byte,
- 16-bit half-word data types – half-words are aligned on 2-byte boundaries

Little Endian Mode Word Data Types



Address

Half word

Byte

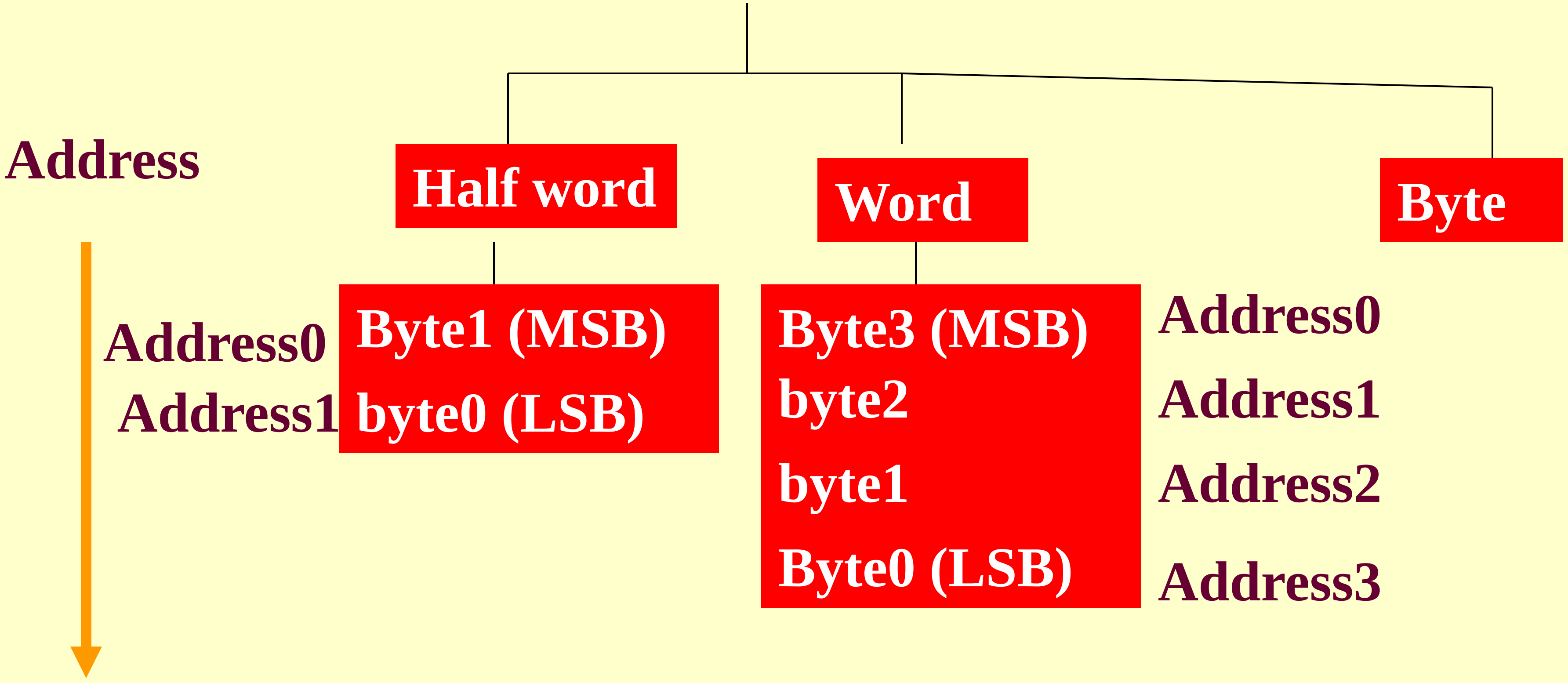
Address0 Byte0 (LSB)

Address1 byte1 (MSB)

Address0- even address; Address1- odd address

Two DataType options

Big Endian Mode Word Data Types



Address

Half word

Address0 Byte1 (MSB)

Address1 byte0 (LSB)

Word

Byte3 (MSB) Address0

byte2 Address1

byte1 Address2

Byte0 (LSB) Address3

Byte

Address0- even address; Address1- odd address

Two DataType options

Programming Model

- 8 general-purpose Lower set of registers permit 3-bit in the instruction for a register
- Program counter register (R15)
- Link Register (R14)
- Always Stack Pointer (R13)

16-bit Thumb Instruction Set Features

- Code size small than 32-bit instruction
- Most instructions - data processing, data transfer and control flow *execute* with no-condition field
- 3-bit for Lower set register facilitates implementation with 16-bit instruction

Instructions Examples-

- LDMIA:Thumb: Store multiple registers
- LDR: Load register (32 bits?)
- LDRB:Load byte (8 bit) into register
- LDRH: Load half-word (!)into register

Instructions Examples-

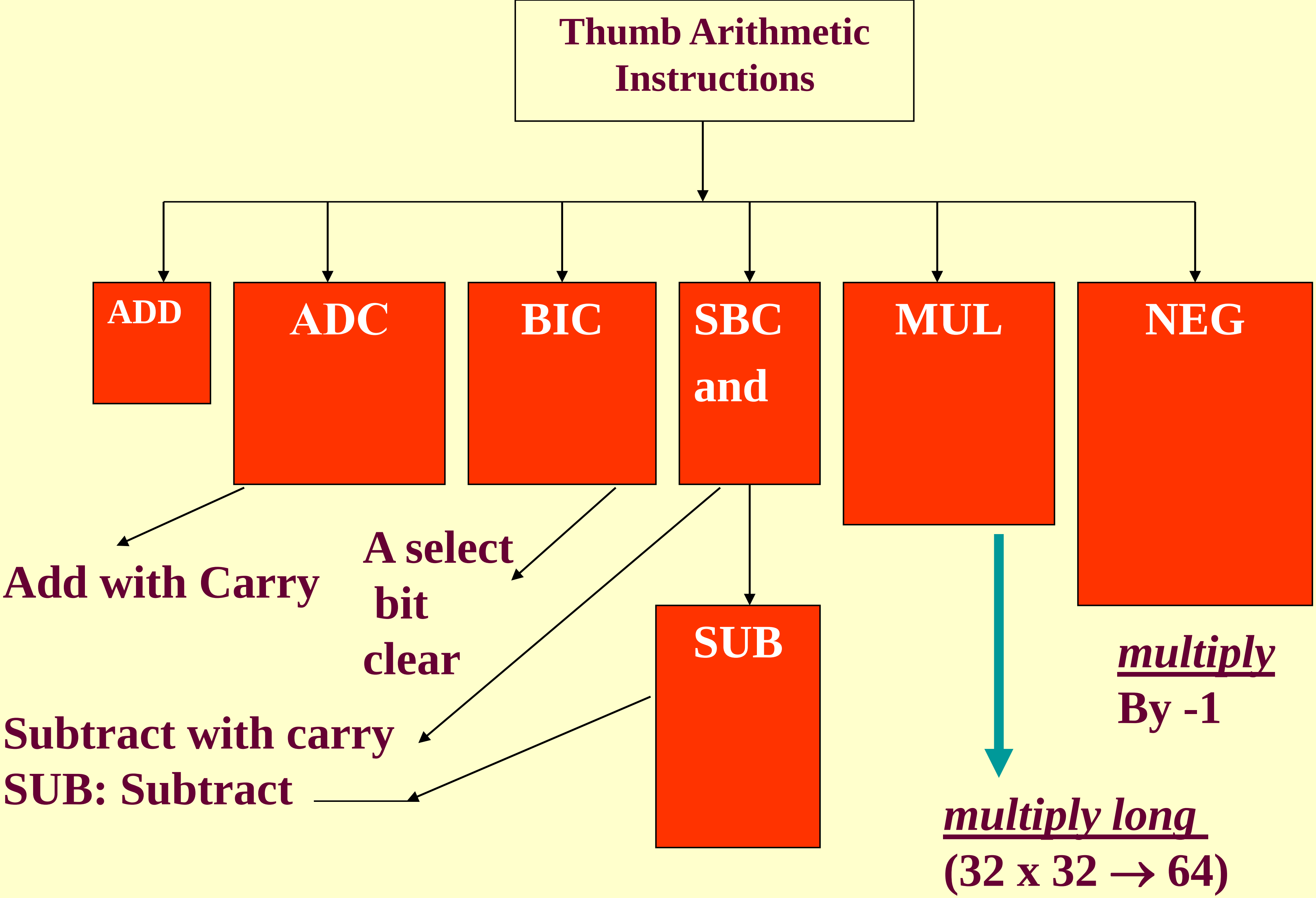
- LDRSB: Load signed byte (sign + 7 bit) into register
- LDRSH: Load signed half-word (sign + 15 bit) into register

Instructions Examples-

- STMIA:Thumb: Store multiple registers
- STR: Store register (32 bit?)
- STRB:Store byte (8 bit)
- STRH: Store half-word (16 bit)

Data Transfer Instructions

- MOV: Move value/register into a register
- MVN: Move after negation
- PUSH: registers to stack
- POP: registers from stack



Thumb Arithmetic Instructions

ADD

ADC

BIC

SBC and

MUL

NEG

SUB

Add with Carry

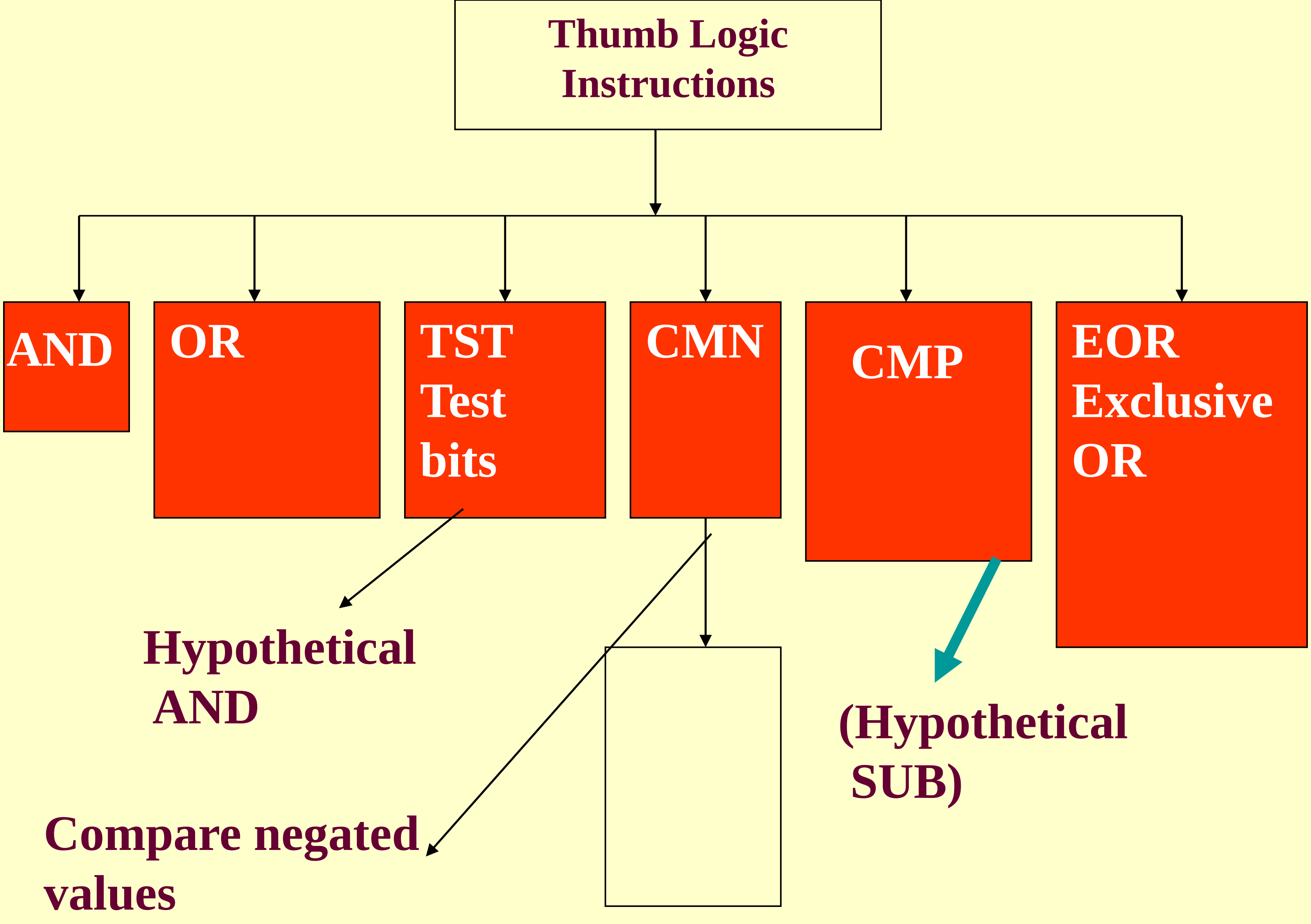
A select bit clear

Subtract with carry

SUB: Subtract

multiply long
(32 x 32 → 64)

multiply
By -1



Thumb Logic Instructions

AND

OR

TST Test bits

CMN

CMP

EOR Exclusive OR

Hypothetical AND

Compare negated values

(Hypothetical SUB)

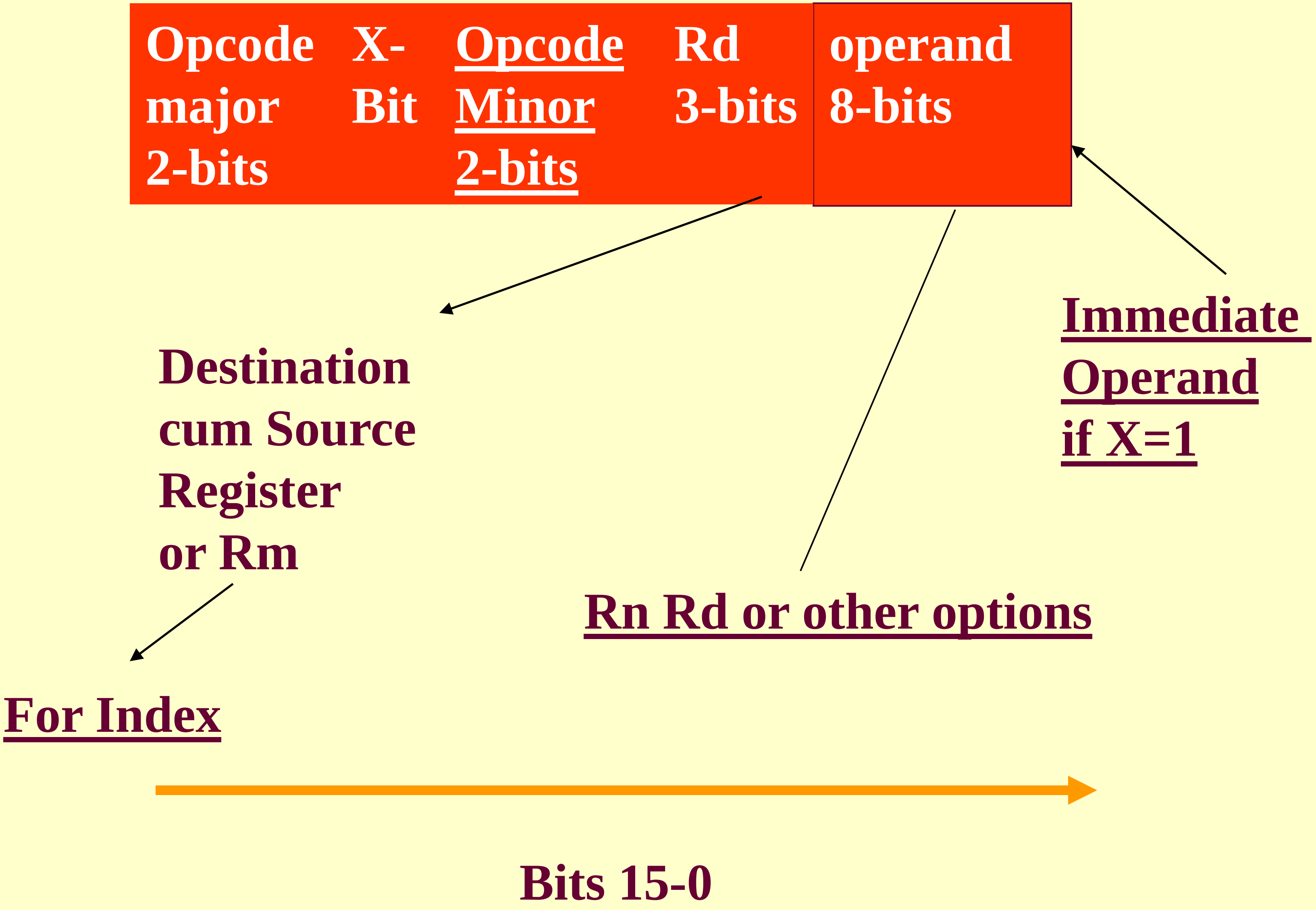
Program Flow Control …

- NOP: No operation,
- B: Branch, BX: Branch and Exchange
- BL: Long Branch with Link (LR and PC exchange), BLX:Branch with Link and Exchange
- BKPT: Breakpoint

Interrupt Control Instruction

- SWI: cond SWI (8 msbs) remaining 8 bit lsbs are irrelevant as far as execution of the instruction is concerned. However, the handler can use the 4-bits for the suffix parameter and a number of 4-bits to reflect the interrupting foreground program (source of SWI).

16-bit Thumb Instruction Format



Opcode major 2-bits	X-Bit	Opcode Minor 2-bits	Rd 3-bits	operand 8-bits

Destination cum Source Register or Rm

For Index

Rn Rd or other options

Immediate Operand if X=1

Bits 15-0

Instruction Example

- ADD r4, #data8 16-bit instruction in memory decompressed at run time into 32-bit format will be 1110 (condition code) 01*1* 01001 (minor opcode) 0 (S bit) 0*100*0 (S bit) 0*100* 0000 (rotation) 1000 0000.

ARM/THUMB Inter-working

- T-bit at CPSR support for inter-working with 16-bit Thumb set and 32-bit ARM. T-bit in CPSR can always switch from ARM to Thumb and vice versa using assembly instruction code16 and code32, respectively.

ARM/THUMB Inter-working

- When working with 16-bit Thumb set execute the instruction code16, T bit set in CPSR. When switching to 32-bit Arm set execute the instruction code32, reset T bit in CPSR.

ARM/THUMB Inter-working

ARM and Thumb code each execute in their own processor state as ARM PC increments by 4 and Thumb by 2

Popular References at Web

http://www.cs.umd.edu/class/fall2001/cmsc411/proj01/arm/home.html

http://www.e-lab.de/ARM7/ARM-instructionset.pdf.

http://www.cvsi.fau.edu/shankar/Presentations/Lecture7_Ch2_ARM.pdf.

http://www.cs.umd.edu/class/fall2001/cmsc411/proj01/arm/thumb.html

http://www.cs.man.ac.uk/Study_subweb/Ugrad/coursenotes/cs1031/Lec20-Thumb.pdf.

Summary

We learnt

- Programmability as little endian and big endian
- ARM 7 - Princeton architecture, ARM 9 - Harvard architecture
- 8-bit byte, 16-bit half-word data types–

We learnt **Thumb®**16 bit subset

Better code density than 32-bit architecture instruction set and switching in-between ARM 32-bit set and Thumb 16-bit set supported

End of Lesson 5 on ARM 16-bit Thumb Instruction Set