ARM Microprocessors 3

The Thumb Instruction Set

- **16-bit Instructions (vs. 32-bit ARM instructions)**
- Used to improve the code density
  - About 30% reduction over ARM for the same code
- Each Thumb instruction mapped to the equivalent ARM instruction:
  - ADD r0, #3 → ADDS r0, r0, #3
- Not conditionally executed except for 'B'
- Separate instructions for the barrel shift operations

Code Size: ARM vs. Thumb

- C-code

```
if (x>=0)     return x;
else          return -x;
```

- ARM assembly version

2 x 4 = 8 bytes

```
iabs    CMP r0,#0       ;Compare r0 to zero
        RSBLT r0,r0,#0  ;If r0<0 (less than=LT) then do r0= 0-r0
        MOV pc,lr       ;Move Link Register to PC (Return)
```

- Thumb assembly version

4 x 3 = 12 bytes

```
        CODE16          ;Directive specifying 16-bit (Thumb)
                        ;instructions
iabs    CMP r0,#0       ;Compare r0 to zero
        BGE return      ;Jump to Return if greater or equal to zero
        NEG r0,r0       ;If not, negate r0
return  MOV pc,lr       ;Move Link register to PC (Return)
```


Thumb-ARM Differences

- Most Thumb instructions executed unconditionally
  - All the ARM instructions executed conditionally
- Many Thumb data processing instructions use a 2-address format (destination reg == one of source reg)
- Less regular instruction formats over ARM (for the code density)

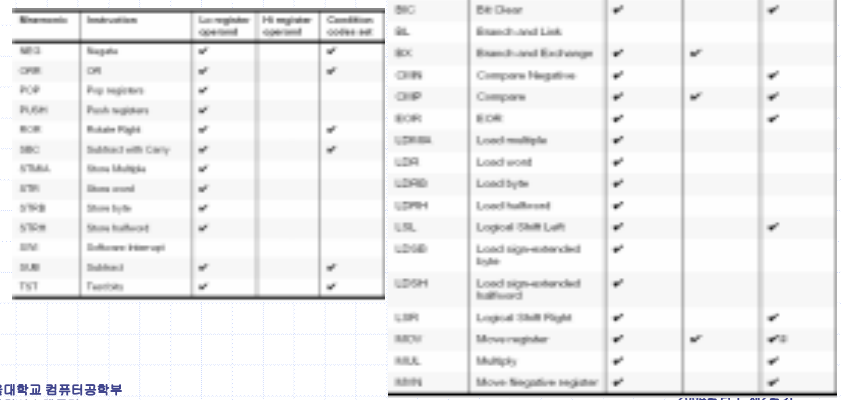
Thumb instruction set

- 19 instruction formats

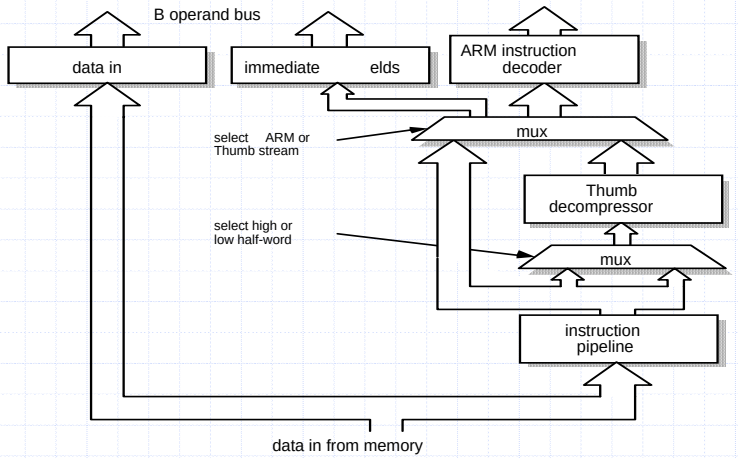
Thumb Register Usage

- r0 ~ r7: fully accessible
- r8 ~ r12: only accessible w / **MOV, ADD, CMP**
- r13, r14, r15: limited accessibility
- cpsr / spsr: no direct access
  - Must switch to ARM state to access cpsr / spsr
  - No coprocessor instructions

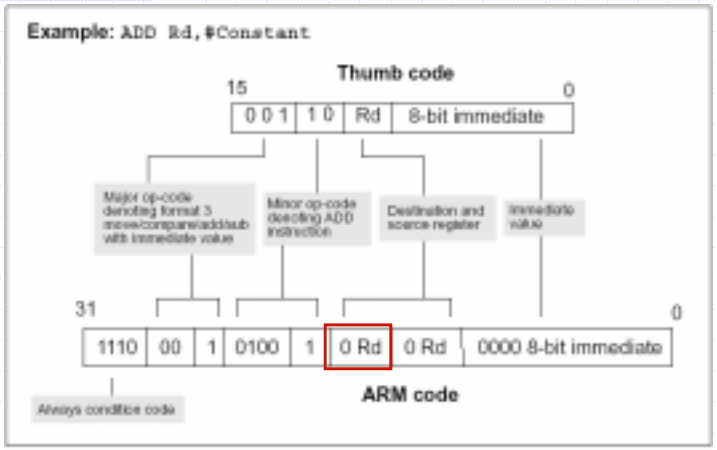
ARM Thumb Instruction Opcodes



Thumb Instruction Decoder Organization



Thumb to ARM Instruction Mapping



Thumb: r0 – r7 only

ARM-THUMB Interworking

- To call a THUMB routine from an ARM routine, the core should switch to 'THUMB' mode:
- T flag in CPSR indicates the current mode.
- **BX** and **BLX** instructions are used to switch ARM/THUMB modes.

BX & BLX Instructions

- **BX** ***Rm*** ; branch exchange
  - **pc** = ***Rm*** & 0xfffffffe
  - T = ***Rm*****[0]**
- **BLX** ***Rm*** **|** ***label*** ; branch exchange w/link
  - **lr** = inst. addr after BLX + T
  - **pc** = ***label, T = label[0]***
  - **pc** = ***Rm & 0xfffffffe, T = Rm[0]***

BLX Example (ARM -> Thumb)

```
        CODE32
        LDR r0, =thumbCode + 1
        BLX r0

        CODE16
thumbCode
        ADD r1, #1
        BX  lr
```

BLX Example (Thumb → ARM)

```
        CODE16
        LDR r0, =ARMCode
        BLX r0

        CODE32
ARMCode
        ADD r1, #1
        BX    lr            ; lr[0] was already set to 1.
```


Thumb Advantages

- Space: About 70% of ARM code
- # of Instructions: About 140% of ARM code
- Exec. Time:
  - With a 32-bit memory, ARM code is about 40% faster over Thumb code
  - With a 16-bit memory, Thumb code is about 45% faster over ARM code
- Thumb code consumes about 30% less memory power.