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

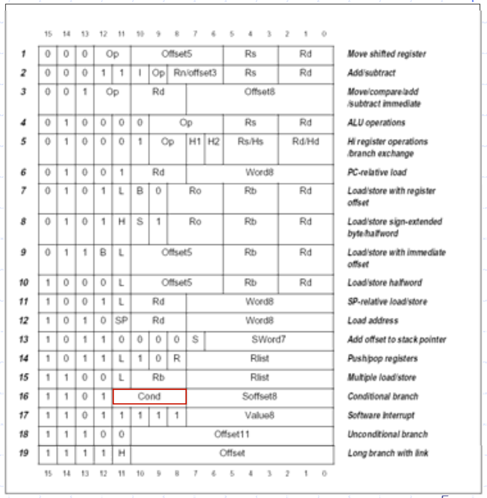
```
          CODE16             ;Directive specifying 16-bit
(Thumb)
                             ;instructions
iabs      CMP r0,#0          ;Compare r0 to zero
          BGE return         ;Jump to Return if greater or equal to zero
          NEG r0,r0          ;If not, negate r0
return    MOV pc,lr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



	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
1	0	0	0	Op		Offset5					Rs			Rd			Move shifted register
2	0	0	0	1	1	I	Op	Rn/offset3			Rs			Rd			Add/subtract
3	0	0	1	Op		Rd			Offset8								Move/compare/add /subtract immediate
4	0	1	0	0	0	0	Op				Rs			Rd			ALU operations
5	0	1	0	0	0	1	Op		H1	H2	Rs/Hs			Rd/Hd			Hi register operations /branch exchange
6	0	1	0	0	1	Rd			Word8								PC-relative load
7	0	1	0	1	L	B	0	Ro			Rb			Rd			Load/store with register offset
8	0	1	0	1	H	S	1	Ro			Rb			Rd			Load/store sign-extended byte/halfword
9	0	1	1	B	L	Offset5					Rb			Rd			Load/store with immediate offset
10	1	0	0	0	L	Offset5					Rb			Rd			Load/store halfword
11	1	0	0	1	L	Rd			Word8								SP-relative load/store
12	1	0	1	0	SP	Rd			Word8								Load address
13	1	0	1	1	0	0	0	0	S	SWord7							Add offset to stack pointer
14	1	0	1	1	L	1	0	R	Rlist								Push/pop registers
15	1	1	0	0	L	Rb			Rlist								Multiple load/store
16	1	1	0	1	Cond				Soffset8								Conditional branch
17	1	1	0	1	1	1	1	1	Value8								Software Interrupt
18	1	1	1	0	0	Offset11											Unconditional branch
19	1	1	1	1	H	Offset											Long branch with link
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	

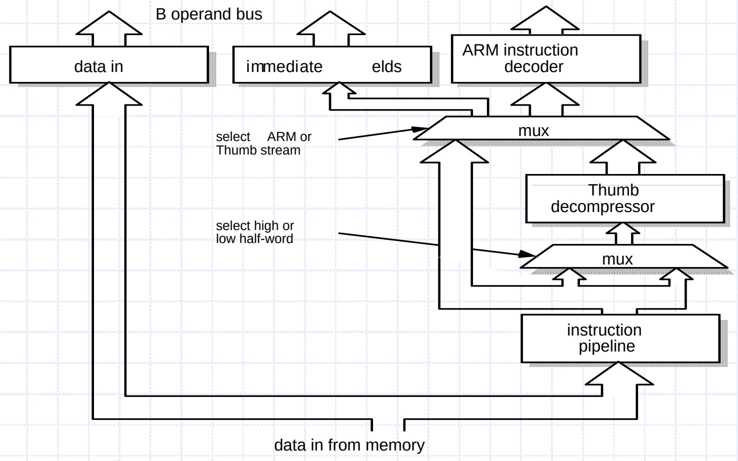
Thumb Register Usage

- 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

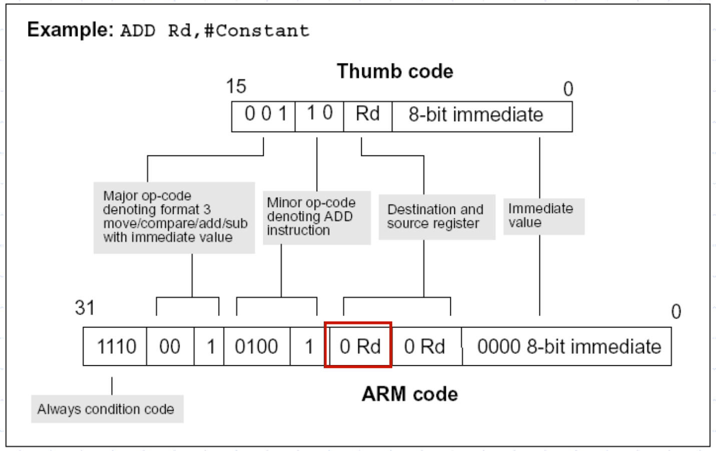
ARM Thumb Instruction Opcodes

Mnemonic	Instruction	Lo register operand	Hi register operand	Condition codes set
ADC	Add with Carry	✔		✔
ADD	Add	✔	✔	✔①
AND	AND	✔		✔
ASR	Arithmetic Shift Right	✔		✔
B	Unconditional branch	✔		
B*xx*	Conditional branch	✔		
BIC	Bit Clear	✔		✔
BL	Branch and Link			
BX	Branch and Exchange	✔	✔	
CMN	Compare Negative	✔		✔
CMP	Compare	✔	✔	✔
EOR	EOR	✔		✔
LDMIA	Load multiple	✔		
LDR	Load word	✔		
LDRB	Load byte	✔		
LDRH	Load halfword	✔		
LSL	Logical Shift Left	✔		✔
LDSB	Load sign-extended byte	✔		
LDSH	Load sign-extended halfword	✔		
LSR	Logical Shift Right	✔		✔
MOV	Move register	✔	✔	✔②
MUL	Multiply	✔		✔
MVN	Move Negative register	✔		✔
NEG	Negate	✔		✔
ORR	OR	✔		✔
POP	Pop registers	✔		
PUSH	Push registers	✔		
ROR	Rotate Right	✔		✔
SBC	Subtract with Carry	✔		✔
STMIA	Store Multiple	✔		
STR	Store word	✔		
STRB	Store byte	✔		
STRH	Store halfword	✔		
SWI	Software Interrupt			
SUB	Subtract	✔		✔
TST	Test bits	✔		✔

Thumb Instruction Decoder Organization



Thumb to ARM Instruction Mapping



Example: ADD Rd,#Constant
Thumb code
15
0
0 0 1
1 0
Rd
8-bit immediate
Major op-code denoting format 3 move/compare/add/sub with immediate value
Minor op-code denoting ADD instruction
Destination and source register
Immediate value
31
0
1110
00
1
0100
1
0 Rd
0 Rd
0000 8-bit immediate
ARM code
Always condition code

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.