

# Branch Instructions

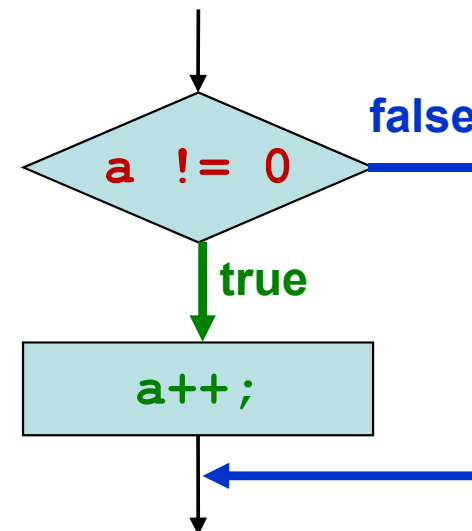
## Computer Engineering 1

**CT Team: A. Gieriet, J. Gruber, B. Koch, M. Loeser, M. Meli,  
M. Rosenthal, M. Ostertag, A. Rüst, J. Scheier**

## ■ Branches may change the PC

- "Non-linear" execution of programs
- Taking decisions

```
uint32_t a;  
  
...  
if (a != 0) {  
    a++;  
}  
...
```



- **Overview Branch Instructions <sup>1)</sup>**
- **Unconditional Branches**
  - B → direct, relative
  - BX → indirect, absolute
- **Conditional Branches**
  - Flag dependent branches
  - Arithmetic branches
    - signed vs. unsigned
- **Compare and Test**
  - **CMP** and **CMN**
  - **TST**

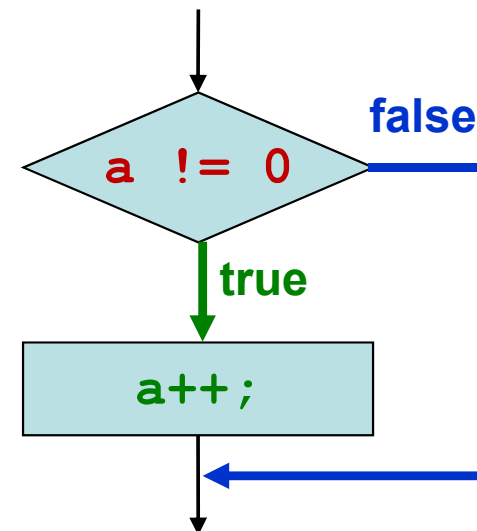
At the end of this lesson you will be able

- to explain what branch instructions are and how they work
- to classify a given branch instruction with regard to
  - conditional / unconditional
  - relative / absolute
  - direct / indirect
- to apply and discuss the different branch instructions
- to determine based on the settings of the flags whether a conditional branch is taken or not
- to distinguish, apply and explain the instructions **CMP**, **CMN** and **TEST**

## ■ Branches may change the PC

- "Non-linear" execution of programs
- Taking decisions

```
uint32_t a;  
  
...  
if (a != 0) {  
    a++;  
}  
...
```

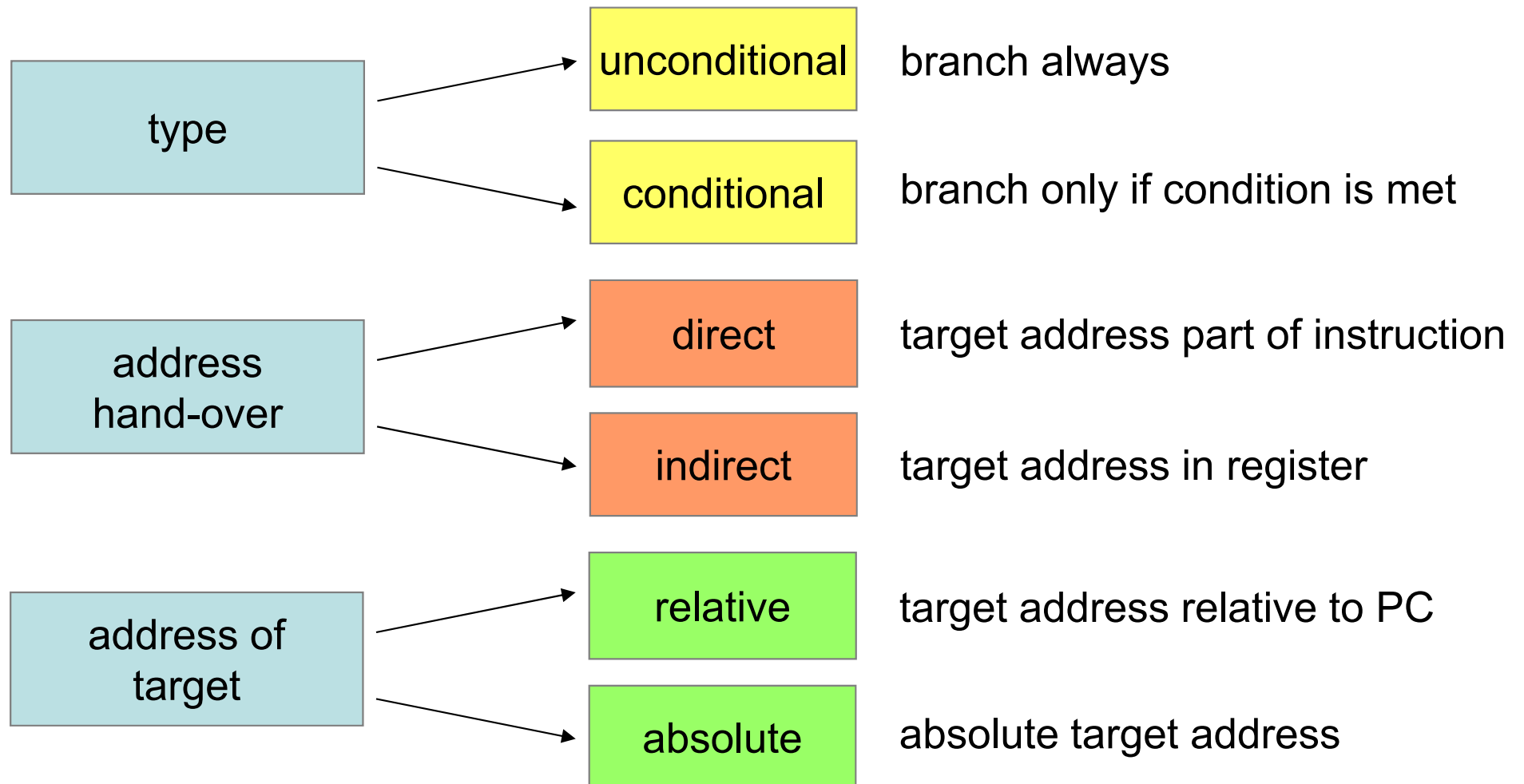


modify PC based  
on decision

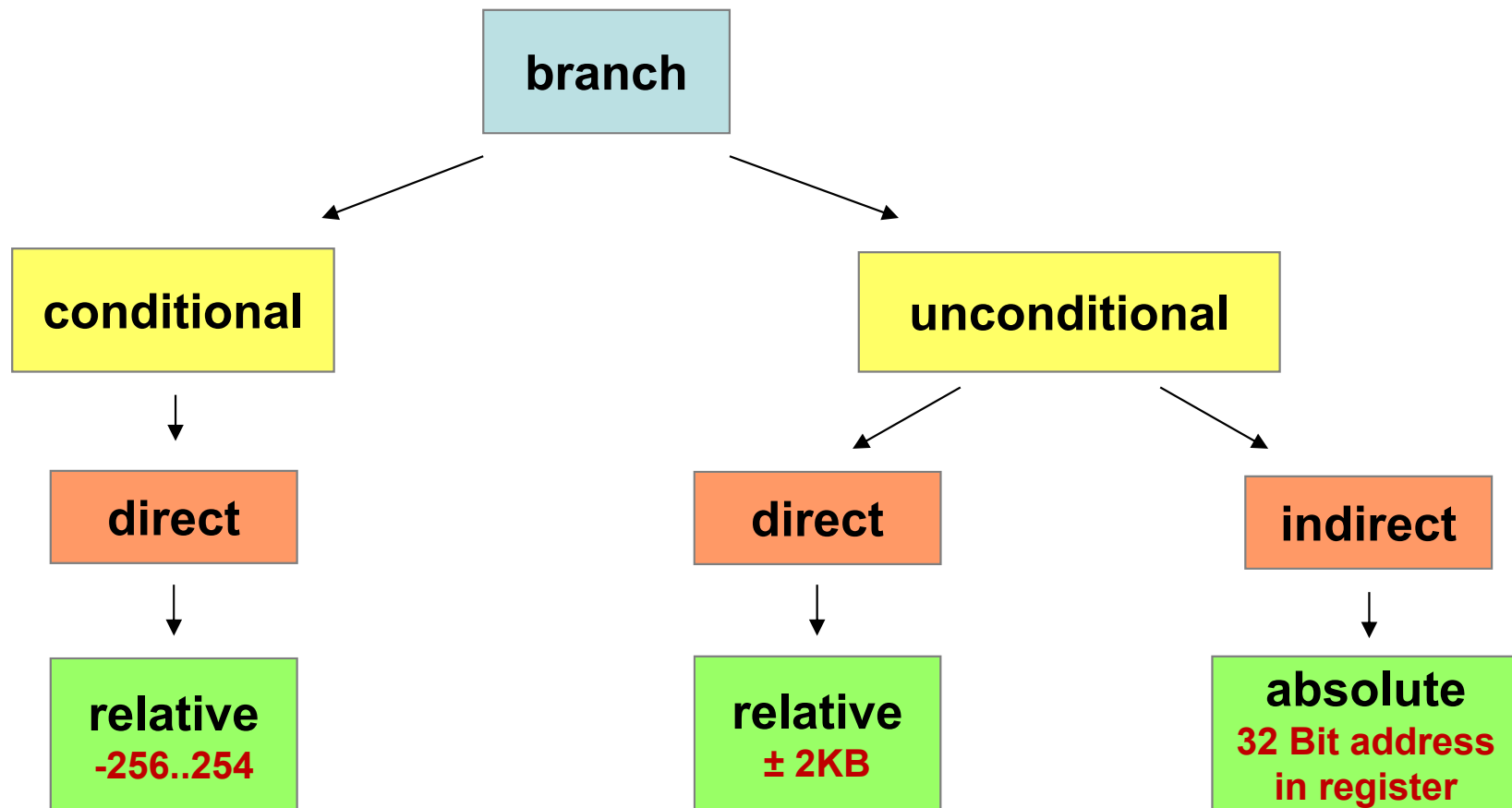
0x20000030	3800	SUBS	r0, #0
0x20000032	d000	BEQ	end_if
0x20000034	1c40	ADDS	r0, r0, #1
0x20000036	... end_if	...	
0x20000038			

# Overview Branch Instructions

## ■ Properties



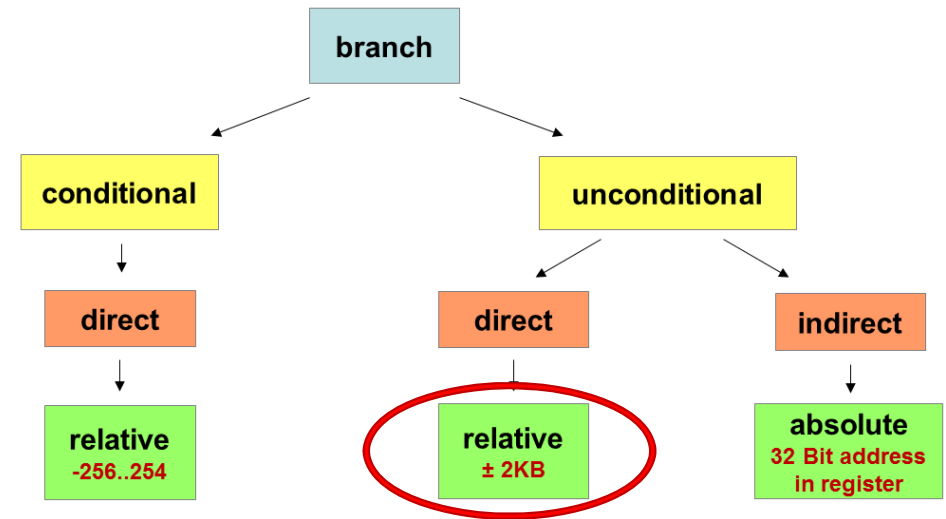
## ■ Overview ARMv6-M (Cortex-M0)



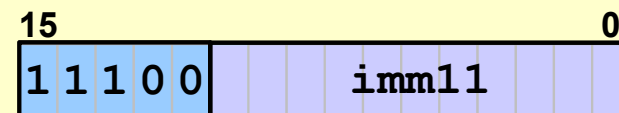
# Unconditional Branches

## ■ B (immediate)

- Unconditional
- Direct
- Relative (to PC)
  - imm11:0
  - Offsets from -2048d to +2046d



B <label>



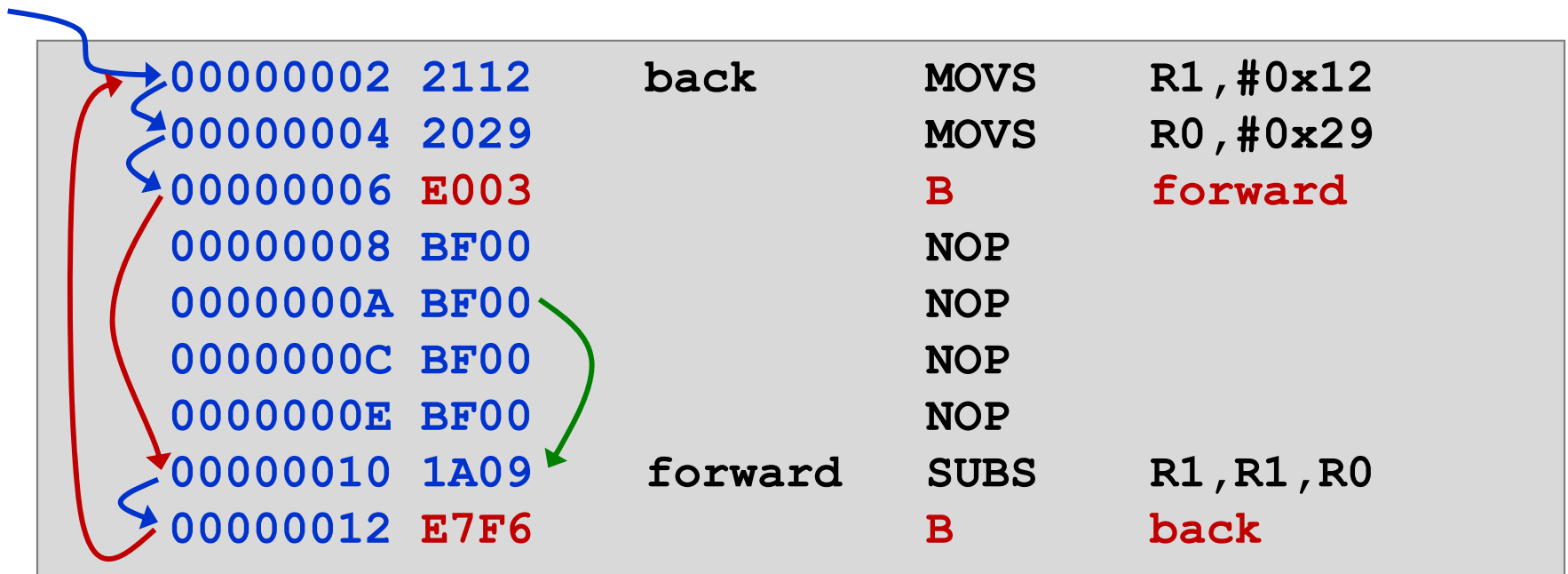
PC = PC + imm11:0



# Unconditional Branches

## ■ Direct, relative branch

→ B label



Forward

$$\begin{array}{rcl}
 0x0000'0006 & \text{Address of current instruction} & \\
 + 0x0000'0004 & \text{1)} & \\
 + 0x0000'0006 & (0x003 \ll 1) = 0x006 & \\
 \hline
 0x0000'0010 & & 
 \end{array}$$

Back

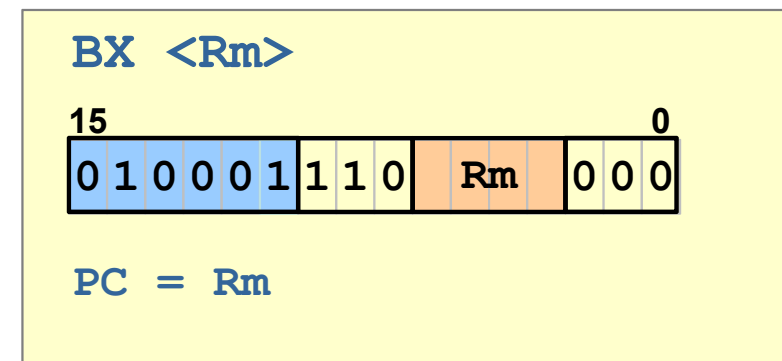
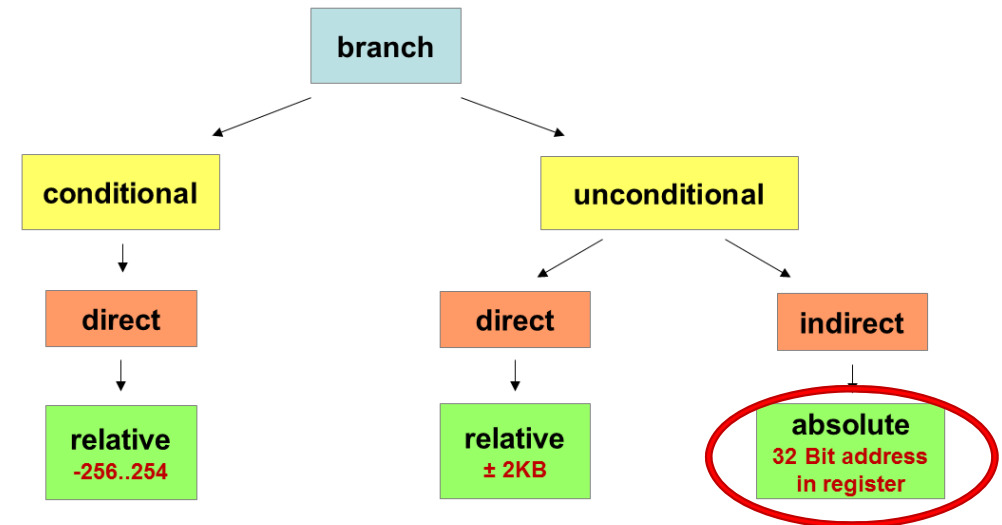
$$\begin{array}{rcl}
 0x0000'0012 & \text{Address of current instruction} & \\
 + 0x0000'0004 & \text{1)} & \\
 + 0xFFFF'FFEC & (0x7F6 \ll 1) = 0xFFEC & \\
 \hline
 0x0000'0002 & = -20d & 
 \end{array}$$

sign-extended

<sup>1)</sup> Because of pipeline, PC is always **current address plus 4**

## ■ BX

- Branch and Exchange
- Register Rm holds target address
- Unconditional
- Indirect
- Absolute



# Unconditional Branches

## ■ Indirect, absolute branch → BX R0

00000014	4802		LDR	R0,=jmpaddr
00000016	4700		BX	R0
00000018	BF00		NOP	
0000001A	BF00		NOP	
0000001C	3013	jmpaddr	ADDS	R0,R0,#0x13
0000001E	BF00		NOP	

- jmpaddr = 0x0000'001C → R0
- R0 → PC

# Conditional Branches

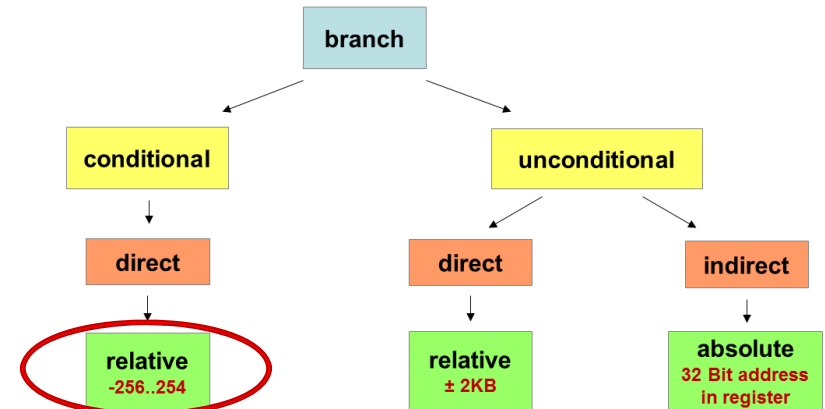
## ■ Flag-dependent branches

- Based on **one specific flag**

## ■ Arithmetic branches

- Based on one or more flags
  - e.g. after an arithmetic instruction
- **unsigned** operations
  - **higher and lower**
- **signed** operations
  - **greater and less**

## ■ Conditional branches are always **relative** on ARM



## ■ Flag-dependent

Symbol	Condition	Flag
EQ	Equal	$Z == 1$
NE	Not equal	$Z == 0$
CS	Carry set	$C == 1$
CC	Carry clear	$C == 0$
MI	Minus/negative	$N == 1$
PL	Plus/positive or zero	$N == 0$
VS	Overflow	$V == 1$
VC	No overflow	$V == 0$

source: Joseph Yiu: *The definite Guide to the ARM Cortex M3*, Page 63

## ■ Arithmetic - unsigned

- higher and lower

Symbol	Condition	Flag
EQ	Equal	$Z == 1$
NE	Not equal	$Z == 0$
HS (=CS)	Unsigned higher or same	$C == 1$
LO (=CC)	Unsigned lower	$C == 0$
HI	Unsigned higher	$C == 1$ and $Z == 0$
LS	Unsigned lower or same	$C == 0$ or $Z == 1$

source: Joseph Yiu: *The definite Guide to the ARM Cortex M3*, Page 63

## ■ Arithmetic - signed

- greater and less

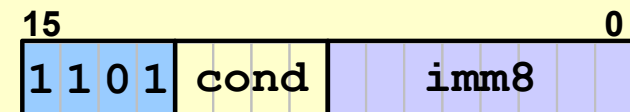
Symbol	Condition	Flag
EQ	Equal	$Z == 1$
NE	Not equal	$Z == 0$
MI	Minus/negative	$N == 1$
PL	Plus/positive or zero	$N == 0$
VS	Overflow	$V == 1$
VC	No overflow	$V == 0$
GE	Signed greater than or equal	$N == V$
LT	Signed less than	$N != V$
GT	Signed greater than	$Z == 0$ and $N == V$
LE	Signed less than or equal	$Z == 1$ or $N != V$

source: Joseph Yiu: The definite Guide to the ARM Cortex M3, Page 63

## ■ Opcodes

- imm8:0
  - Offset from -256d to +254d

B<c> <label>



if (cond) then  
PC = PC + imm8:0

cond	short	Flag
0000	EQ	Z == 1
0001	NE	Z == 0
0010	CS/HS	C == 1
0011	CC/LO	C == 0
0100	MI	N == 1
0101	PL	N == 0
0110	VS	V == 1
0111	VC	V == 0

cond	short	Flag
1000	HI	C == 1 and Z == 0
1001	LS	C == 0 or Z == 1
1010	GE	N == V
1011	LT	N != V
1100	GT	Z == 0 and N == V
1101	LE	Z == 1 or N != V
1110	AL	always
1111	--	--



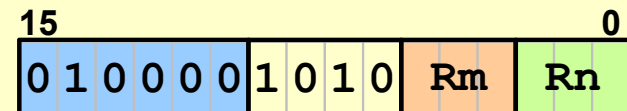
# Compare and Test

## ■ CMP

- Same as **SUBS**, but without storing a result!
- Compare 2 operands
  - Higher/lower?
  - Greater/less?
  - Equal?
- **Only flags are affected!**
- Registers unchanged
- T2 also higher registers

CMP <Rn>, <Rm>

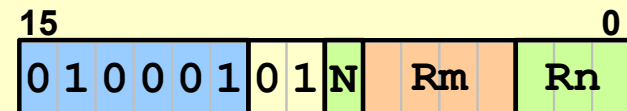
T1



$Rn - Rm \rightarrow N, Z, C, V$

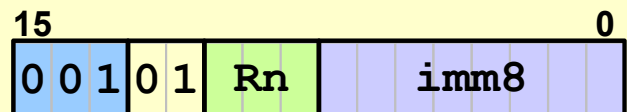
CMP <Rn>, <Rm>

T2



$Rn - Rm \rightarrow N, Z, C, V$

CMP <Rn>, #<imm8>

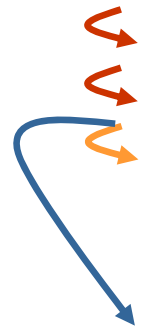


$Rn - \text{<imm8>} \rightarrow N, Z, C, V$

# Compare and Test

## ■ CMP

- **CMP** does not change registers
- Example



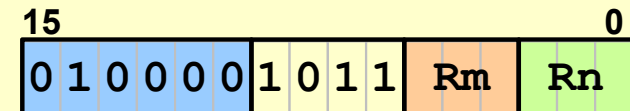
00000000	4288	CMP	R0,R1	; R0 > R1 ?
00000002	D802	BHI	go_on	; if higher -> go_on
00000004	000A	MOVS	R2,R1	; otherwise exchange regs
00000006	0001	MOVS	R1,R0	
00000008	0010	MOVS	R0,R2	
0000000A	2305	go_on	MOVS	R3,#5
0000000C	...	...		

# Compare and Test

## ■ CMN

- Same as **ADDS**, but without storing result!
- Compare 2 operands negative
- Only flags are affected!
- Registers unchanged
- Read **CMN** as
  - Is content of Rm equal to 2's complement of Rn?

CMN <Rn>, <Rm>



$Rn + Rm \rightarrow N, Z, C, V$

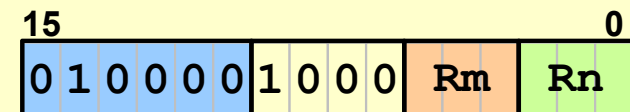
00000000	42C8	CMN	R0, R1	; R0 equal minus R1?
00000002	D002	BEQ	next	

# Compare and Test

## ■ TST

- Is a specific bit set?
- Logical AND without storing result
- Registers unchanged
- Changes only flags N and Z
  - C and V unchanged

**TST <Rn>, <Rm>**



**Rn & Rm → N, Z**

```
SWITCH_ADDRESS      EQU      0x60000200
S3_MASK             EQU      0x00000008
00000000 4903       LDR      R1,=SWITCH_ADDRESS
00000002 6808       LDR      R0,[R1]           ; read switch data
00000004 4A03       LDR      R2,=S3_MASK
00000006 4210       TST      R0,R2           ; bit S3 = 1 ?
00000008 D101       BNE      s3_equal_one    ; branch if Z = 0
0000000A ...       s3_equal_zero
...
...               s3_equal_one
...
```

## ■ Which branches are taken?

Instruction	Z	C	N	V	“Taken” / “Not Taken”?
BNE label	1	0	0	0	
BLO label	0	0	0	0	
BHI label	0	1	0	0	
BLT label	0	0	1	1	
BLE label	1	0	1	1	

“Taken”

“Not Taken”

Execution is continued at the indicated label

Execution is continued at the instruction following the branch instruction

## ■ Branch Instructions Change **PC**

- „Decision Making“ and Control Flow

## ■ Branch Instructions

- unconditional, relative, direct      **B**       $PC = PC \pm 2KB$
- unconditional, absolute, indirect      **BX**       $PC = Rm$
- conditional, relative, direct      **Bxx**       $PC = PC-256; PC+254$

## ■ Compare and Test

- **CMP**, **CMN**      → **SUBS**, **ADDS** without result, but flags are set!
- **TST**      → **AND** without result, but flags are set!