Registers

	N	Z	C	V	
cpsr	1	0	0	1	

r0	0x40000000 (2^{30})
r1	0x40000000 (2^{30})
r2	0x80000000 (-2^{31})
r3	

```
adds r2, r1, r0
```

Note the **s**!

What will the **cpsr** be after?

A: N=0, Z=0, C=0, V=1
B: N=0, Z=1, C=0, V=1
C: N=1, Z=0, C=1, V=1
D: N=1, Z=0, C=0, V=1
E: Something else

Instruction Set

1. **adds**: Perform an addition on the two sources, store it in the destination register, and set the **Current Process Status Register**:
   - N set to 1 if the result was *Negative*
   - Z set to 1 if the result was exactly *Zero*
   - C set to 1 if the result would unsigned *"carry"*
   - V set to 1 if the result would signed *"oVerflow"*

```
adds r2, r1, r0
```

Note the **s**!

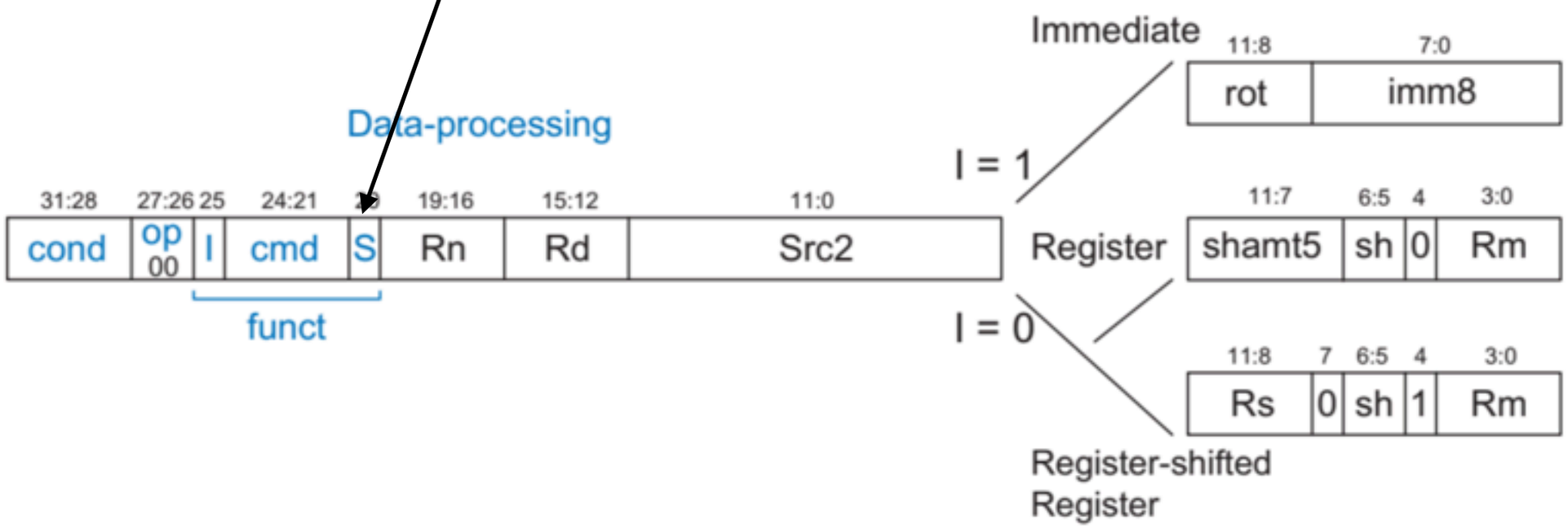


Figure B.1 Data-processing instruction encodings

Harris et al, *Digital Design and Computer Architecture, Second Edition*. 2016. Appendix B

Registers

	N	Z	C	V	
cpsr	1	0	0	1	

r0	0x40000000 (2^{30})
r1	0x40000000 (2^{30})
r2	0x80000000 (-2^{31})
r3	

Instruction Set

1. **movvs**: Move a value to a register **if the V status bit is 1**. If it is 0, do nothing. The **vs** part is a suffix that can be used on most instructions.

```
adds r2, r1, r0     @ if it overflowed,
movvs r2, #0        @ set value to 0
```

Table 6.3 Condition mnemonics

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS/HS	Carry set / unsigned higher or same	C
0011	CC/LO	Carry clear / unsigned lower	$\overline{C}$
0100	MI	Minus / negative	N
0101	PL	Plus / positive or zero	$\overline{N}$
0110	VS	Overflow / overflow set	V
0111	VC	No overflow / overflow clear	$\overline{V}$
1000	HI	Unsigned higher	$\overline{Z}C$
1001	LS	Unsigned lower or same	Z OR $\overline{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\overline{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	Z OR $(N \oplus V)$
1110	AL (or none)	Always / unconditional	Ignored

The parts of the status register that are checked are given in this table.

The suffix on the instruction corresponds to the "cond" part

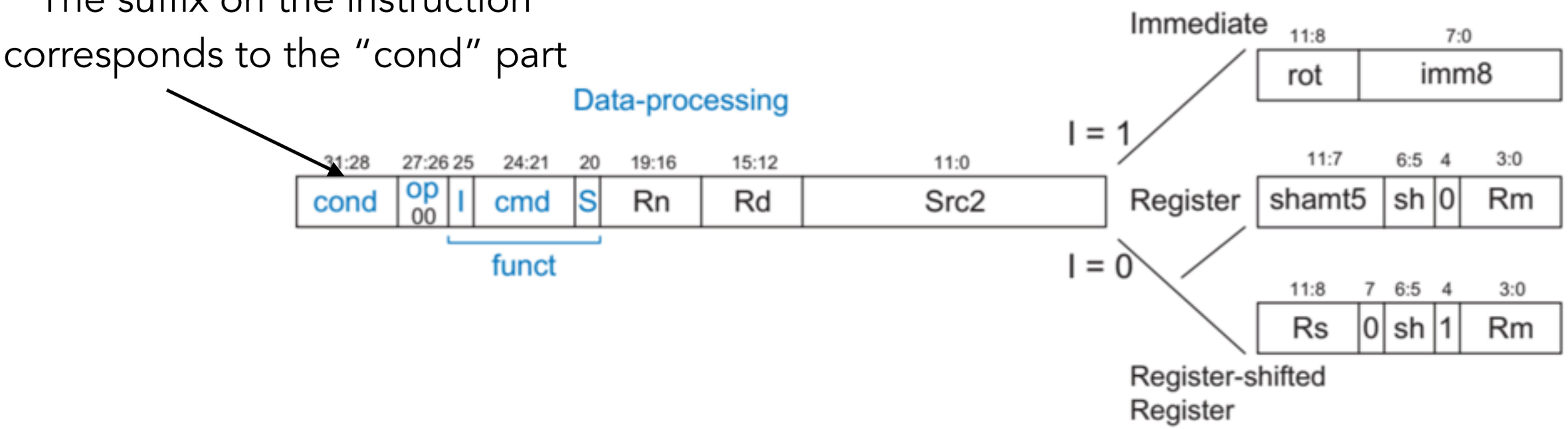


Figure B.1 Data-processing instruction encodings

Registers

	N	Z	C	V	
cpsr	1	0	0	1	

r0	0x00000000 (0)
r1	0x7FFFFFFF (2^{31} - 1)
r2	
r3	

Instruction Set

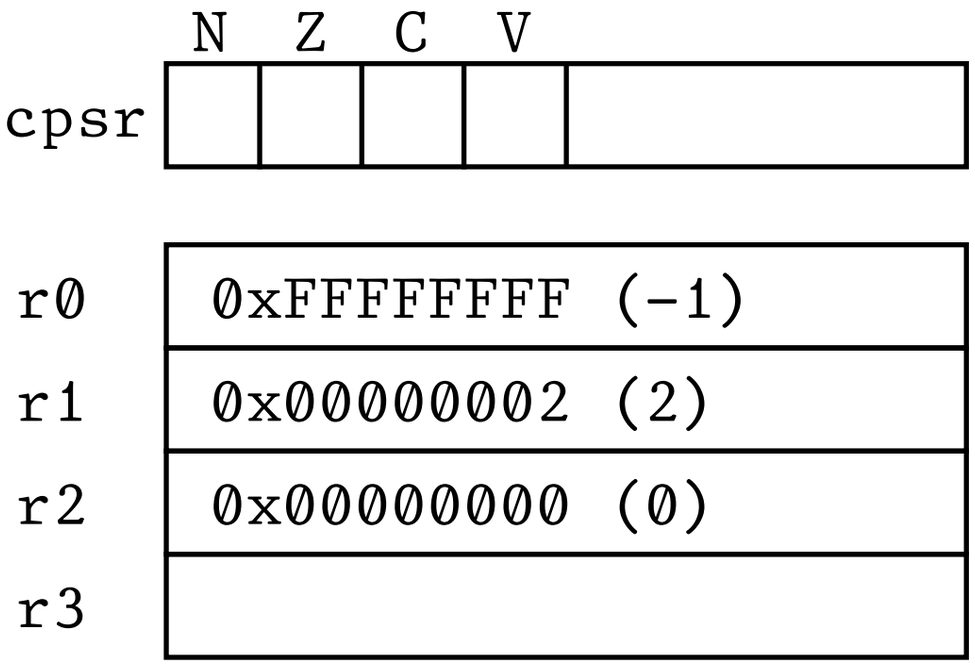
1. **movvs**: Move a value to a register **if the V status bit is 1**. If it is 0, do nothing. The **vs** part is a suffix that can be used on most instructions.

```
adds r1, r1, #1     @ if it overflowed,
addvs r0, #1        @ set value to 0, and
movvs r1, #0        @ increment r0
```

What is in **r0**, **r1** after this runs?

A: r0 = 0, r1 = 1
B: r0 = 1, r1 = 1
C: r0 = 0, r1 = 0x80000000
D: r0 = 1, r1 = 0x80000000
E: Something else

Registers




```
adds r0, r1, r0
movne r3, #0
```

What is in **`r3`** after this runs?

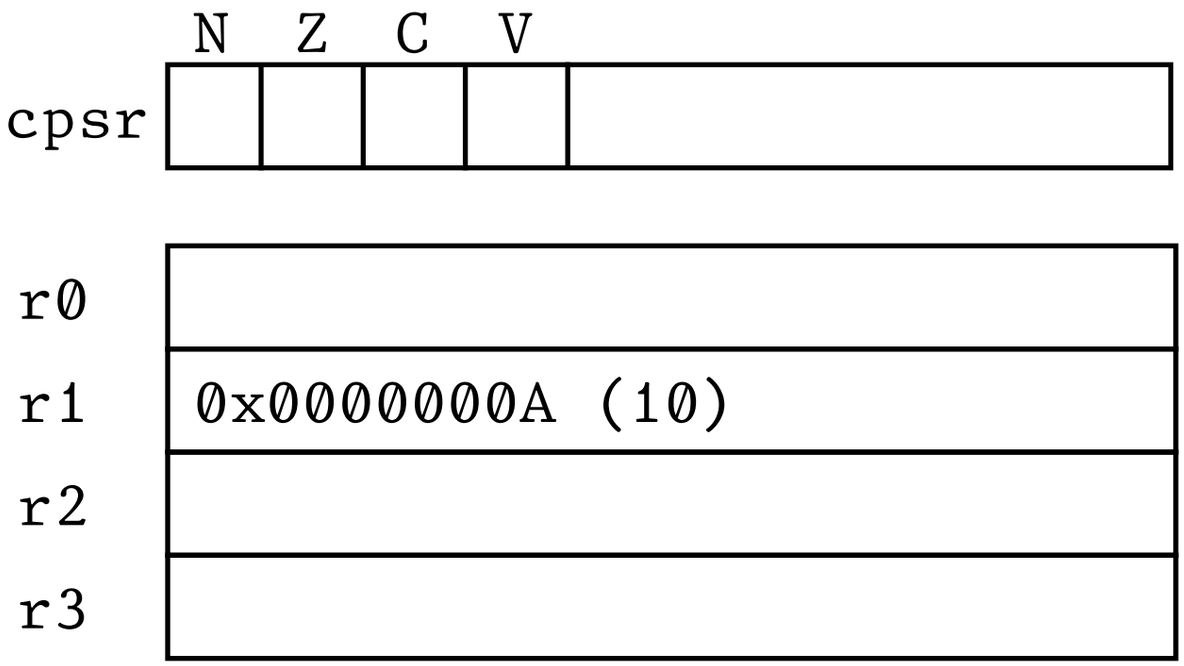
Instruction Set

Table 6.3 Condition mnemonics

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS/HS	Carry set / unsigned higher or same	C
0011	CC/LO	Carry clear / unsigned lower	$\overline{C}$
0100	MI	Minus / negative	N
0101	PL	Plus / positive or zero	$\overline{N}$
0110	VS	Overflow / overflow set	V
0111	VC	No overflow / overflow clear	$\overline{V}$
1000	HI	Unsigned higher	$\overline{Z}C$
1001	LS	Unsigned lower or same	Z OR $\overline{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\overline{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	Z OR $(N \oplus V)$
1110	AL (or none)	Always / unconditional	Ignored

Compute the sum from 0
to the value stored in r1,
put the answer in r0.

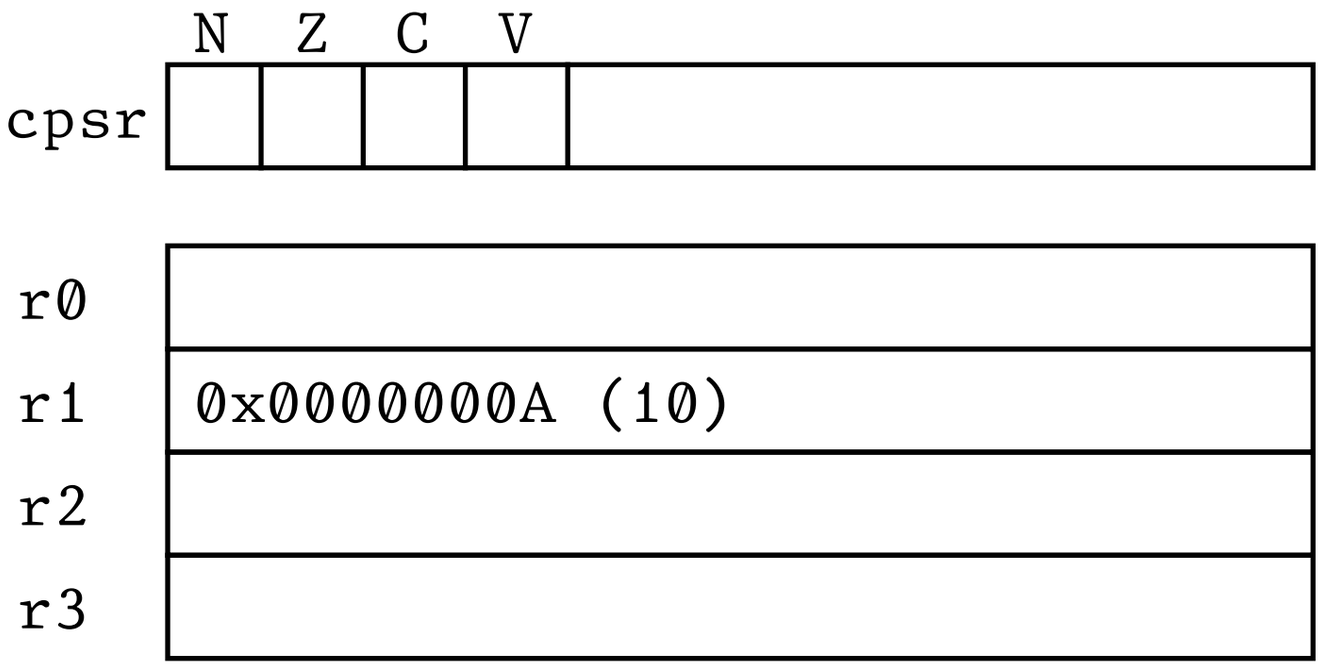
Registers




```
mov r0, #0
add r0, r1, r0
subs r1, #1
@go back and loop if r1 is non-zero!
```

Compute the sum from 0
to the value stored in r1,

Registers



The **Program Counter** determines which instruction to run next. It's "just" another register.

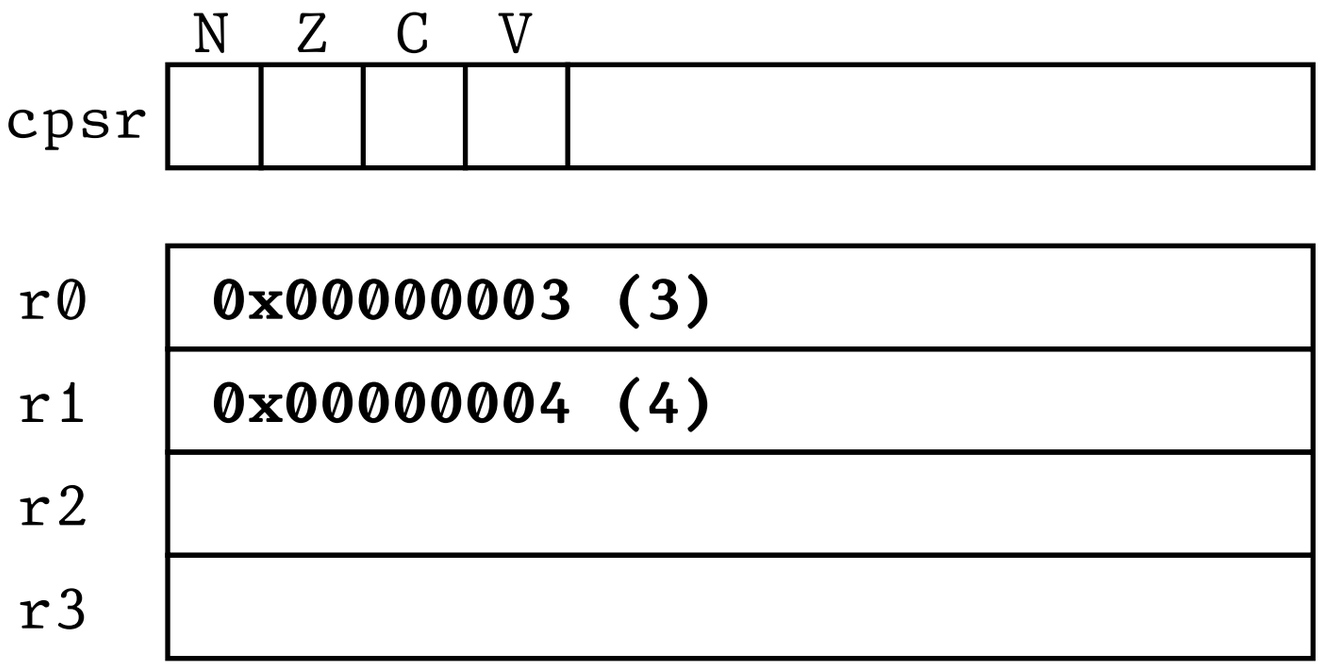
```
0x100 mov r0, #0
0x104 add r0, r1, r0
0x108 subs r1, #1
0x10c
```

```
@go back three steps and loop!
```

Each instruction is at an address

Compute the sum from 0 to the value stored in r1,

Registers




```
0x100 mov r2, #1
0x104 mul r2, r2, r1
0x108 subs r0, #1
0x10c subne r15, #8
```

Each instruction is at an address

What is in r2 after the program runs?

A: 0x0000000c (12)

B: 0x00000051 (81)

C: 0x00000027 (27)

D: 0x00000003 (3)

Write a program that subtracts 7 from r1 until r1 is below zero, and stores the number of subtractions in r2.

Table 6.3 Condition mnemonics

cond	Mnemonic	Name	CondEx
0000	EQ	Equal	Z
0001	NE	Not equal	$\overline{Z}$
0010	CS/HS	Carry set / unsigned higher or same	C
0011	CC/LO	Carry clear / unsigned lower	$\overline{C}$
0100	MI	Minus / negative	N
0101	PL	Plus / positive or zero	$\overline{N}$
0110	VS	Overflow / overflow set	V
0111	VC	No overflow / overflow clear	$\overline{V}$
1000	HI	Unsigned higher	$\overline{Z}C$
1001	LS	Unsigned lower or same	Z OR $\overline{C}$
1010	GE	Signed greater than or equal	$\overline{N \oplus V}$
1011	LT	Signed less than	$N \oplus V$
1100	GT	Signed greater than	$\overline{Z}(\overline{N \oplus V})$
1101	LE	Signed less than or equal	Z OR $(N \oplus V)$
1110	AL (or none)	Always / unconditional	Ignored