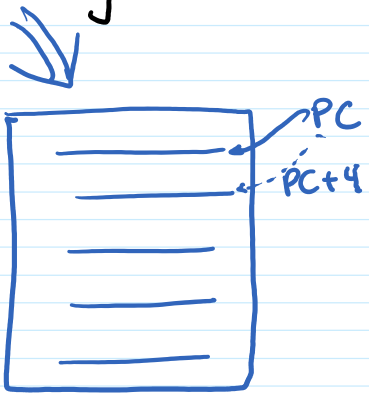
This section → Control flow instructions
→ Branch and link instructions
→ Conditional execution instructions

Control Flow Instructions

- These instructions change the order of instruction execution
  - ↳ Normally flow is sequential execution
  - ↳ PC is incremented by 4 after executing every instruction



- Types of conditional flow instructions:
  1. ↳ unconditional branch
  2. ↳ conditional branch
  3. ↳ Branch and link
  4. ↳ conditional execution
- We will look at these instructions one by one:

1 Unconditional branch instruction

```
        B       Target
        ...
        ...
Target  ...
```

2 Conditional branch instruction

```
        MOV     r2,#0
LOOP    ...
        ...
        ADD     r2,r2,#1
        CMP     r2,#20
        BNE     LOOP
        ...
```

Conditional Branch instructions:

Conditional Branch instructions:

• BEQ, BNE	equal or not equal to
• BPL, PMI	Result positive or negative
• BCC, BCS	Carry clear or set
• BVC, BVS	Overflow clear or set
• BGT, BGE	Greater than, greater than or equal to
• BLT, BLE	cd

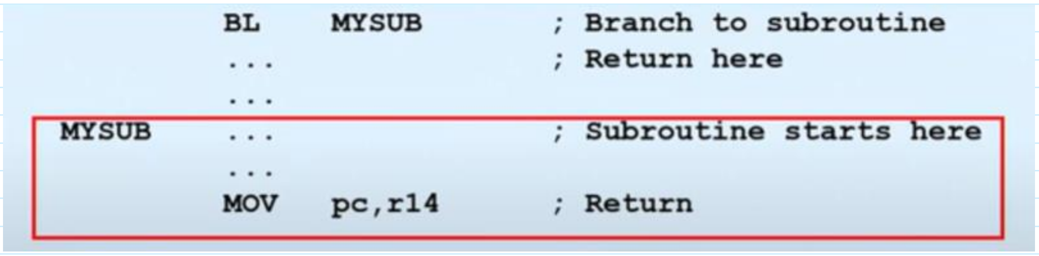
③ Branch and Link Instructions

↳ Used for calling subroutines in ARM

→ The return address is saved in register r14 (link register)

→ To return from the subroutine, we have to jump back to the address stored in r14.

Simple example

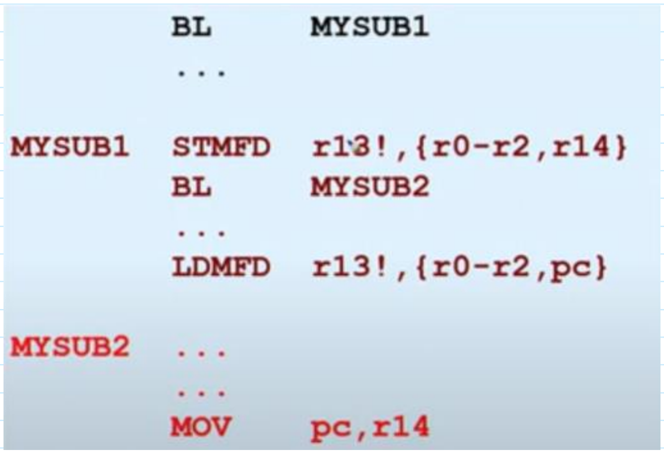


```
          BL      MYSUB          ; Branch to subroutine
          ...                    ; Return here
          ...
MYSUB     ...                    ; Subroutine starts here
          ...
          MOV     pc,r14         ; Return
```

* Nested subroutine calls cannot be used in this way *

↳ ∴ we must use a software stack to save/restore the return address and registers.

→ Here is an example using nested subroutine calls and return



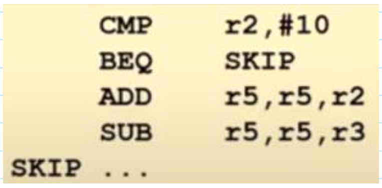
```
        BL      MYSUB1
        ...

MYSUB1  STMFD   r13!,{r0-r2,r14}
        BL      MYSUB2
        ...
        LDMFD   r13!,{r0-r2,pc}

MYSUB2  ...
        ...
        MOV     pc,r14
```

* With the above, nested subroutine calls can be used.

4 Conditional Execution

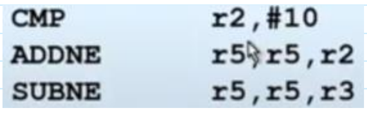
- a unique feature of the ARM instruction set
- all instructions can be made conditional
  ↳ ie will get executed only when a specified condition is true.
- Helps remove many short instructions
  ↳ improves performance and code density.
- Example

  if (r2 != 10) r5 = r5+10 −r3



```
        CMP     r2,#10
        BEQ     SKIP
        ADD     r5,r5,r2
        SUB     r5,r5,r3
SKIP    ...
```

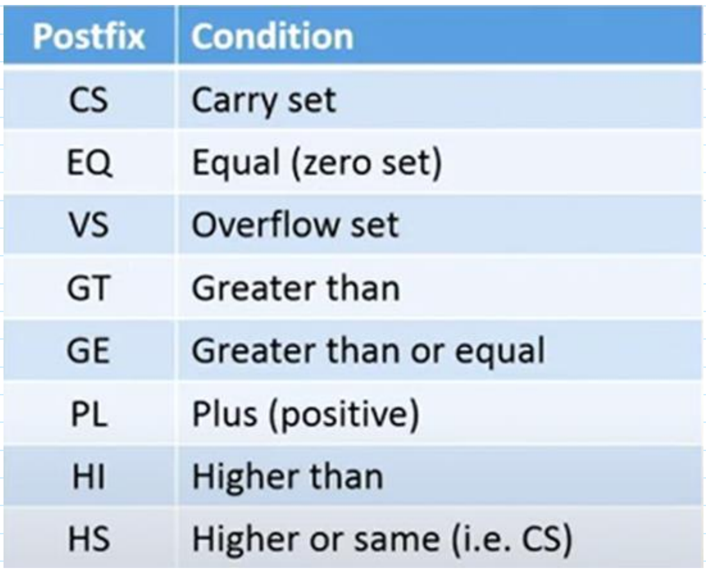
Conventional Approach



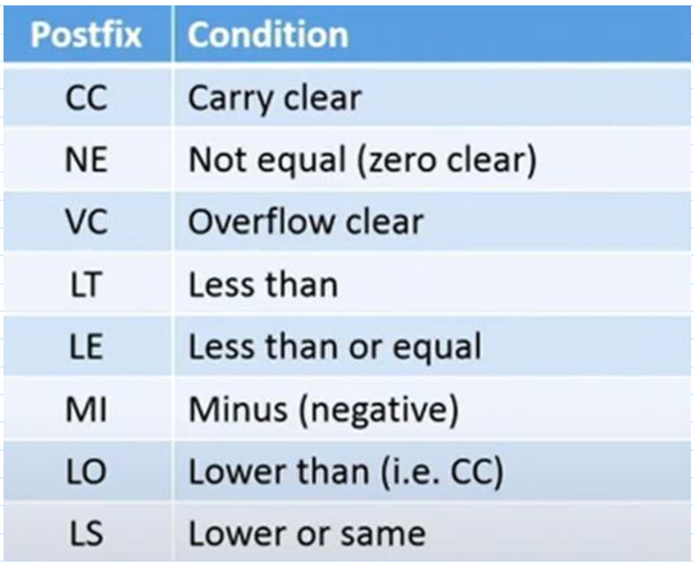
```
CMP         r2,#10
ADDNE       r5,r5,r2
SUBNE       r5,r5,r3
```

ARM Approach

- Various instruction postfix are supported for conditional execution



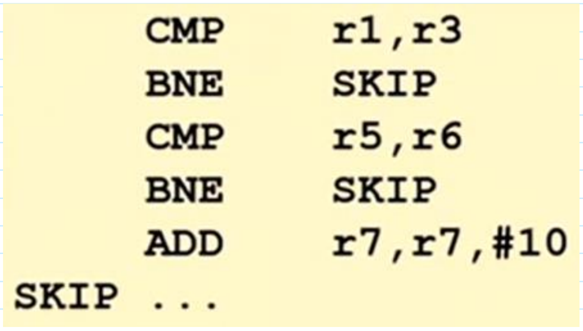
Postfix	Condition
CS	Carry set
EQ	Equal (zero set)
VS	Overflow set
GT	Greater than
GE	Greater than or equal
PL	Plus (positive)
HI	Higher than
HS	Higher or same (i.e. CS)



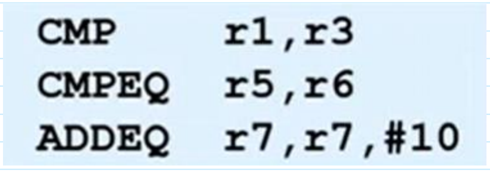
Postfix	Condition
CC	Carry clear
NE	Not equal (zero clear)
VC	Overflow clear
LT	Less than
LE	Less than or equal
MI	Minus (negative)
LO	Lower than (i.e. CC)
LS	Lower or same

Another example:

if ((r1 ==r3)) && (r5 ==r6) r7=r7+10



```
        CMP     r1,r3
        BNE     SKIP
        CMP     r5,r6
        BNE     SKIP
        ADD     r7,r7,#10
SKIP    ...
```



```
CMP     r1,r3
CMPEQ   r5,r6
ADDEQ   r7,r7,#10
```

Supervisor Calls

- The software Interrupt Instruction (SWI) is used to enter Supervisor Mode, usually to request a particular supervisor function.
- The CPSR is saved into the Supervisory Mode SPSR and execution branches to the SWI vector.
- The SWI handler reads the opcode to extract the SWI function number.