Branches

P. Bernardi, R. Ferrero

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Unconditional branch

- There are four instructions for unconditional branch:
  - branch `B <label>`
  - branch indirect `BX <Rn>`
  - branch and link `BL <label>`
  - branch indirect with link `BLX <Rn>`
- `BL` and `BLX` save the return address (i.e., the address of the next instruction) in `LR` (`r14`) and they are used to call subroutines.

Branch range

- In the `B` instruction, the opcode is 8 bit and the immediate value is 24 bit.
- Since addresses are halfword-aligned, the immediate value specifies bit 24-1 of the relative address.
- The 25th is for the sign; so the relative address can be $\pm 2^{24}$ byte = $\pm$16 MB.
- `BX` can jump to any 32-bit value = 4 GB.

MOV for unconditional branch

- `B` and `BX` change the value of `PC`.
- Similarly, a jump can be implemented by changing the value of `PC` with `MOV` and `LDR`:
  - ```
    LDR <Rd>, =<label>
    MOV PC, <Rd>
    ```
  - ```
    LDR PC, =<label>
    ```
- `MOV` and `LDR` force the last bit of `PC` to 0.
- `MOV` instead of `BX` is discouraged: the assembler generates a warning.

Conditional branch: B?? and BX??

??	Flags	Meaning	??	Flags	Meaning
EQ	Z = 1	equal	NE	Z = 0	not equal
CS HS	C = 1	unsigned ≥	CC LO	C = 0	unsigned <
MI	N = 1	negative	PL	N = 0	positive or 0
VS	V = 1	overflow	VC	V = 0	no overflow
HI	C =1 & Z = 0	unsigned >	LS	C =0 & Z = 1	unsigned ≤
GE	N ≥ V	signed ≥	LT	N ≠ V	signed <
GT	Z =0 or N = V	signed >	LE	Z =1 or N ≠ V	signed ≤

Example: do you pass the exam?

```
; r0 contains the score of the exam
        CMP r0, #18
        BEQ refuse
        BLO reject
        BHI accept
        ...
refuse  ...     ; study more
reject  ...     ; study much more
accept  ...     ; go on holiday
```

For loop

The pseudocode of the for loop is

```
for (i = 0; i < N; i++) {
    ...        //do something
}
```

For loop: naive implementation

```
        MOV r0, #0
loop    CMP r0, #N
        BHS exit
        ...         ; do something
        ADD r0, r0, #1
        B loop
exit
```

For loop: optimization

```
        MOV r0, N
loop    ...         ; do something
        SUBS r0, r0, #1
        BNE loop
exit
```

Do-While loop

The pseudocode

```
do {
    ...         //do something
} while (r0 != N);
```

can be implemented as:

```
loop    ...         ; do something
test    CMP r0, #N
        BNE loop
```