Control Flow Instructions

CSE 30: Computer Organization and Systems Programming

Diba Mirza
Dept. of Computer Science and Engineering
University of California, San Diego
So Far...

- All instructions have allowed us to manipulate data
- So we’ve built a calculator
- In order to build a computer, we need ability to change the flow of control in a program…
Ways to change Control flow in C

1. goto <label>
2. if (condition) { //do something }
3. if-else
4. Loops
   I. do-while
   II. for
   III. while
   IV. switch
Labels

- Any instruction can be associated with a label
- Example:
  
  ```c
  start    ADD r0,r1,r2 ; a = b+c
  next     SUB r1,r1,#1 ; b--
  ```

- In fact, every instruction has a label regardless if the programmer explicitly names it
  - The label is the address of the instruction
  - A label is a pointer to the instruction in memory
  - Therefore, the text label doesn’t exist in binary code
ARM goto Instruction

- The simplest control instruction is equivalent to a C goto statement
- goto label (in C) is the same as:
- B label (in ARM)
- B is shorthand for “branch”. This is called an unconditional branch meaning that the branch is done regardless of any conditions.
- There are also conditional branches:
Conditional Branch

- To perform a conditional branch,
  1. First set the condition bits (N,Z,V,C) in the program status register
  2. Then check on these condition bits to branch conditionally

- What are the ways we have learnt to set condition bits so far?
  - Append S to arithmetic/logical instruction
  - There is another way using a ‘Comparison Instruction’
Comparison Instructions

- **CMP** – Compare and set condition bits
  - subtracts a register or an immediate value from a register value and updates condition codes
  - Unlike SUB, it doesn’t store the result anywhere

- **Examples:**
  - CMP r3, #0 ; set Z flag if r3 == 0
  - CMP r3, r4 ; set Z flag if r3 == r4

All flags are set as result of this operation, not just Z

Conditional branches often preceded by CMP
ARM Decision Instructions

❖ ARM has variants of the branch instruction that only goto the label if a certain condition is TRUE

❖ Examples:
  ❖ BEQ label ; BRANCH EQUAL
  ❖ BNE label ; BRANCH NOT EQUAL
  ❖ BLE label ; BRANCH LESS THAN EQUAL
  ❖ BLT label ; BRANCH LESS THAN
  ❖ BGE label ; BRANCH GREATER THAN EQUAL
  ❖ BGT label ; BRANCH GREATER THAN
  ❖ Plus more …

❖ The condition is T/F based upon the fields in the Program Status Register
The possible condition codes are listed below:

- Note AL is the default and does not need to be specified.

<table>
<thead>
<tr>
<th>Suffix</th>
<th>Description</th>
<th>Flags tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>Equal</td>
<td>Z=1</td>
</tr>
<tr>
<td>NE</td>
<td>Not equal</td>
<td>Z=0</td>
</tr>
<tr>
<td>CS/HS</td>
<td>Unsigned higher or same</td>
<td>C=1</td>
</tr>
<tr>
<td>CC/LO</td>
<td>Unsigned lower</td>
<td>C=0</td>
</tr>
<tr>
<td>MI</td>
<td>Minus</td>
<td>N=1</td>
</tr>
<tr>
<td>PL</td>
<td>Positive or Zero</td>
<td>N=0</td>
</tr>
<tr>
<td>VS</td>
<td>Overflow</td>
<td>V=1</td>
</tr>
<tr>
<td>VC</td>
<td>No overflow</td>
<td>V=0</td>
</tr>
<tr>
<td>HI</td>
<td>Unsigned higher</td>
<td>C=1 &amp; Z=0</td>
</tr>
<tr>
<td>LS</td>
<td>Unsigned lower or same</td>
<td>C=0 or Z=1</td>
</tr>
<tr>
<td>GE</td>
<td>Greater or equal</td>
<td>N=V</td>
</tr>
<tr>
<td>LT</td>
<td>Less than</td>
<td>N!=V</td>
</tr>
<tr>
<td>GT</td>
<td>Greater than</td>
<td>Z=0 &amp; N=V</td>
</tr>
<tr>
<td>LE</td>
<td>Less than or equal</td>
<td>Z=1 or N=!V</td>
</tr>
<tr>
<td>AL</td>
<td>Always</td>
<td></td>
</tr>
</tbody>
</table>
C Code

\[
\text{If}(X == 0) \\
X = Y + Z;
\]

Assume \(X\), \(Y\), and \(Z\) are integers in registers \(r0\), \(r1\), and \(r2\), respectively.

Q: Which one is the equivalent assembly code?

A

```
CMP r0, #0
BEQ Label
ADD r0, r1, r2
Label:
```

B

```
CMP r0, #0
BNE Label
ADD r0, r1, r2
Label:
```

C – Neither of these is correct.
Assume $X$, $Y$, and $Z$ are integers in registers $r0$, $r1$, and $r2$, respectively.

Q: Which is the equivalent assembly code?

A: 
```
CMP r0, #0
BEQ Label
ADD r0, r1, r2
Label:
```

B: 
```
CMP r0, #0
BNE Label
ADD r0, r1, r2
Label:
```

C – Neither of these is correct.
C if–else

- if statements in C
  - if (condition) {clause}
  - if (condition) {clause1} else {clause2}

- Rearrange if–else into following:
  ```
  if (condition) goto L1;
  clause2;
  goto End;
  L1: clause1;
  End:
  ```

- Not as elegant as if-else, but same meaning
- Now let’s try to write equivalent ARM code
Compiling C if into ARM

- Compile by hand
  ```c
  if (i == j) f=g+h;
  else f=g-h;
  ```

- Use this mapping:
  - f: r0
  - g: r1
  - h: r2
  - i: r3
  - j: r4
Compiling C if into ARM

Compile by hand

```c
if (r3 == r4) r0=r1+r2;
else r0=r1-r2;
```

- Final compiled ARM code:
  ```
  CMP r3, r4 ; Z = 1 if i==j
  BEQ True ; goto True when i==j
  SUB r0, r1, r2 ; f=g-h(false)
  B Fin ; goto Fin
  True ADD r0, r1, r2 ; f=g+h (true)
  Fin
  ```

Note: Compiler automatically creates labels to handle decisions (branches) appropriately. Generally not found in C code.
Loops in C/Assembly

- There are three types of loops in C:
  - `while`
  - `do... while`
  - `for`
Loops in C/Assembly

- Simple loop in C;
  ```c
  do {
      g--;  
      i = i + j;
  } while (i != h);
  ```

- Rewrite this as:
  ```c
  Loop: g--;  
  i = i + j;
  if (i != h) goto Loop;
  ```

- Use this mapping:
  ```
  g: r1, h: r2, i: r3, j: r4
  ```
Loops in C/Assembly

- Replace variables with equivalent registers

  Loop:     r1--;  
             r3 = r3 + r4;  
             if (r3 != r2) goto Loop;

- Final compiled ARM code:

  Loop   SUB r1,r1,#1 ; r1--  
           ADD r3,r3,r4 ; r3=r3+r4  
           CMP r3,r2 ; cmp r3,r2  
           BNE Loop ; goto Loop if r3!=r2
Inequalities in ARM

- Until now, we’ve only tested equalities (== and != in C). General programs need to test < and > as well.
- Use CMP and BLE, BLT, BGE, BGT
- Examples:
  - if (f < 10) goto Loop; => CMP r0,#10
    BLT Loop
  - if (f >= i) goto Loop; => CMP r0,r3
    BGE Loop
- Try on your own:
  - for(i=0;i<20;i++) {Statements}
  - while(x<30) {Statements}
Choose among four alternatives depending on whether \( k \) has the value 0, 1, 2 or 3. Compile this C code:

```c
switch (k) {
    case 0: f=i+j; break; /* k=0*/
    case 1: f=g+h; break; /* k=1*/
    case 2: f=g–h; break; /* k=2*/
    case 3: f=i–j; break; /* k=3*/
}
```
Example: The C Switch Statement

- This is complicated, so simplify.
- Rewrite it as a chain of if-else statements, which we already know how to compile:

  ```c
  if (k==0) f=i+j;
  else if (k==1) f=g+h;
  else if (k==2) f=g-h;
  else if (k==3) f=i-j;
  ```

- Use this mapping:

  ```
  f: r0, g: r1, h: r2, i: r3, j: r4, k: r5
  ```
Example: The C Switch Statement

```c
C code:
If(r5==0) r0=r3+r4;
else if(r5==1) r0=r1+r2;
else if(r5==2) r0=r1-r2;
else if(r5==3) r0=r3-r4;
```
Summary

- A Decision allows us to decide which pieces of code to execute at run-time rather than at compile-time.
- C Decisions are made using conditional statements within an if, while, do while or for.
- CMP instruction sets status register bits.
- ARM Decision making instructions are the conditional branches: BNE, BEQ, BLE, BLT, BGE, BGT.
Conclusion

- Instructions so far:
  - Previously:
    - ADD, SUB, MUL, MULA, [U|S]MULL, [U|S]MLAL, RSB
    - AND, ORR, EOR, BIC
    - MOV, MVN
    - LSL, LSR, ASR, ROR
    - LDR, LDR, STR, LDRB, STRB, LDRH, STRH
  - New:
    - CMP, B{EQ, NE, LT, LE, GT, GE}