CSE 30: Computer Organization and Systems Programming

Lecture 13: Control Transfer Instructions

Diba Mirza
University of California, San Diego
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:

  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 (cpsr)
  2. Then check on these condition bits to branch conditionally

Condition bits:
N (Negative) = 1 if result is negative
Z (Zero) = 1 if result is zero
V (Overflow) = 1 if overflow
C (carry out) = 1 if carry out of MSB
Conditional Branch

• How can we set the condition bits in the cpsr register?
  – Append S to arithmetic/logical instruction, OR
  – Use a ‘Comparison Instruction’

\[
\begin{align*}
\text{SUBS } & \quad r0, r1, r2 \quad @ \quad r0 = r1 - r2 \\
\text{ADD S } & \quad r0, r1, r2 \\
\text{CMP } & \quad r1, r2 \quad @
\end{align*}
\]

\[
\begin{align*}
\text{SUBS } & \quad r0, r1, r2 \quad @ \quad r0 = r1 - r2, \text{ result is not stored but condition bits are set} \\
\text{CMP } & \quad r1, r2 \quad @
\end{align*}
\]

\[
\begin{align*}
N & \quad Z & \quad V & \quad C \\
0 & \quad 1 & \quad 0 & \quad 1, \text{ if } r1 = r2 \\
1 & \quad 0 & \quad 0 & \quad 0, \text{ if } r1 < r2
\end{align*}
\]
Comparison Instructions

• CMP — Compare and set condition bits
  – subtracts a register or an immediate value from a register value and updates condition bits
  – 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 are often preceded by CMP
Recall: 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

• How can we check on condition bits?

\[
\begin{align*}
\text{CMP} & \quad \text{ro, #4} \\
\text{BEQ} & \quad \text{label @ Branch if equal} \\
\text{ADD} & \quad \text{r4, r5, r6 @ if ro! = 4, add is executed} \\
\text{SUB} & \quad \text{ro, ro, #1}
\end{align*}
\]
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 \( i f \ r_0 = r_1 \)
  - BNE label ; BRANCH NOT EQUAL \( i f \ r_0 \neq r_1 \)
  - BLE label ; BRANCH LESS THAN EQUAL \( r_0 \leq r_1 \)
  - BLT label ; BRANCH LESS THAN \( r_0 < r_1 \)
  - BGE label ; BRANCH GREATER THAN EQUAL \( r_0 \geq r_1 \)
  - BGT label ; BRANCH GREATER THAN \( r_0 > r_1 \)
  - Plus more ...

• The condition is T/F based upon the fields in the Program Status Register
ARM Condition codes

• If a condition code is appended to an instruction with a condition code, the instruction is conditionally executed by checking the flags in the CPSR register.

  Condition     Flags tested
  – EQ : Z=1
  – NE : Z=0
  – LE : Z=1 or N=!V
  – LT : N=!V
  – GE : N=V
  – GT : Z=0 & N=V
  – Plus more ...

• The condition is T/F based upon the fields in the Program Status (CPSR) Register.
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.
if (x==y)
  x=x+y;
else
  x=x-y;
While loops

while (a<0)
    a++;

Mapping
a: r0

Loop:
    CMP r0, #0
    BGE end
    AND r0, r0, #1
    B loop

end:

Loop:
    CMP r0, #0
    ADDLT r0, r0, #1
    BLT loop
For loops

for (i=0; i<10; i++){
    a++;
    b--;
}

loop:
    mov r0, #0
    cmp r0, #10
    Bge end
    add r1, r1, #1
    sub r2, r2, #1
    add r0, r0, #1
    B loop

end:
    mov r0, #0
    cmp r0, #10
    addlt r1, r1, #1
    sublt r2, r2, #1
    addlt r0, r0, #1
    blt loop

i: r0
a: r1
b: r2