Topic 7: Control Flow Instructions

CSE 30: Computer Organization and Systems Programming
Summer Session II

Dr. Ali Irturk
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 make decisions…

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

UCSD

C Decisions: `if` Statements

❖ `if` statements in C

   ❖ `if` (*condition*) *clause*

   ❖ `if` (*condition*) *clause1* `else` *clause2*

❖ Rearrange 2nd `if` into following:

```
if (condition) goto L1;
    clause2;
    goto L2;
 L1: clause1;
 L2:
```

   ❖ Not as elegant as if-else, but same meaning

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

ARM Decision Instructions

❖ ARM also 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

Program Status Registers

31		28	27		24	23							16	15				8	7	6	5	4		0
N	Z C V		Q			J	n	d	e	f	i	n	e	n	e	d			I	F	T		mode	
f						s								x					c					
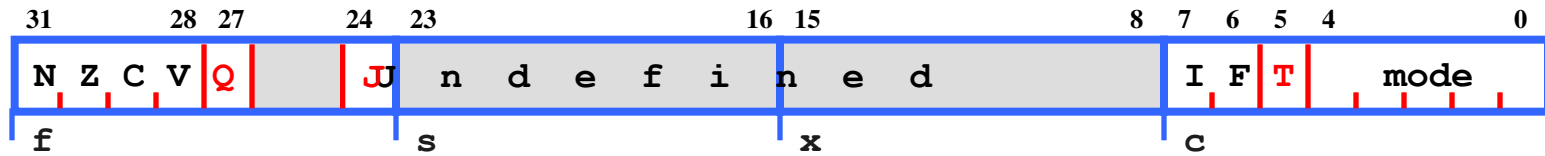
- ❖ Condition code flags
  - ❖ N = **N**egative result from ALU
  - ❖ Z = **Z**ero result from ALU
  - ❖ C = ALU operation **C**arried out
  - ❖ V = ALU operation o**V**erflowed
- ❖ Sticky Overflow flag - Q flag
  - ❖ Architecture 5TE/J only
  - ❖ Indicates if saturation has occurred
- ❖ J bit
  - ❖ Architecture 5TEJ only
  - ❖ J = 1: Processor in Jazelle state

- ❖ Interrupt Disable bits.
  - ❖ I = 1: Disables the IRQ.
  - ❖ F = 1: Disables the FIQ.
- ❖ T Bit
  - ❖ Architecture xT only
  - ❖ T = 0: Processor in ARM state
  - ❖ T = 1: Processor in Thumb state
- ❖ Mode bits
  - ❖ Specify the processor mode

Flags and Their Use

- ❖ The N flag
  - ❖ Set if the result is negative or equivalently if the MSB == '1'
- ❖ The Z flag
  - ❖ Set if the result is zero
- ❖ The C flag
  - ❖ Set if
    - ❖ The result of an addition is greater than 2^{32}
    - ❖ The result of a subtraction is positive
    - ❖ Carryout from the shifter is '1'
- ❖ The V flag (oVerflow)
  - ❖ Set if there is overflow

Condition Codes

❖ The possible condition codes are listed below

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

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

The ARM Register Set

Only need to worry about cpsr (current program status register)

Current Visible Registers

Abort Mode

| r0 |
| r1 |
| r2 |
| r3 |
| r4 |
| r5 |
| r6 |
| r7 |
| r8 |
| r9 |
| r10 |
| r11 |
| r12 |
| r13 (sp) |
| r14 (lr) |
| r15 (pc) |

| cpsr |
| spsr |

Banked out Registers

User	FIQ	IRQ	SVC	Undef
	r8			
	r9			
	r10			
	r11			
	r12			
r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)	r13 (sp)
r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)	r14 (lr)

| | spsr | spsr | spsr | spsr |

UCSD

Compiling C `if` into ARM

❖Compile by hand

```
if (i == j) f=g+h;
else f=g-h;
```


(true)
i == j

i == j?

(false)
i != j

f=g+h   **f=g-h**

Exit

❖Use this mapping:

f: r0, g: r1, h: r2, i: r3, j: r4
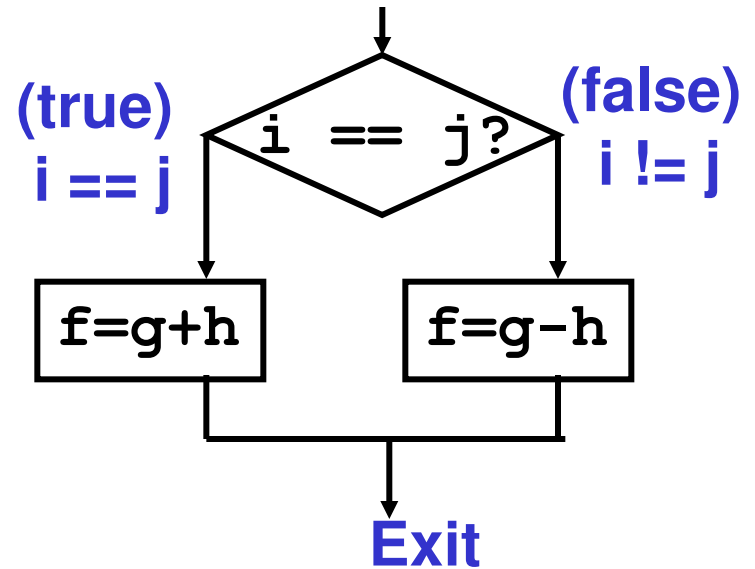
Comparison Instructions

❖ In order to perform branch on the "==" operation we need a new instruction

❖ `CMP` – Compare: subtracts a register or an immediate value from a register value and updates condition codes

❖ 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.

Compiling C `if` into ARM

Compile by hand

```
if (i == j) f=g+h;
else f=g-h;
```
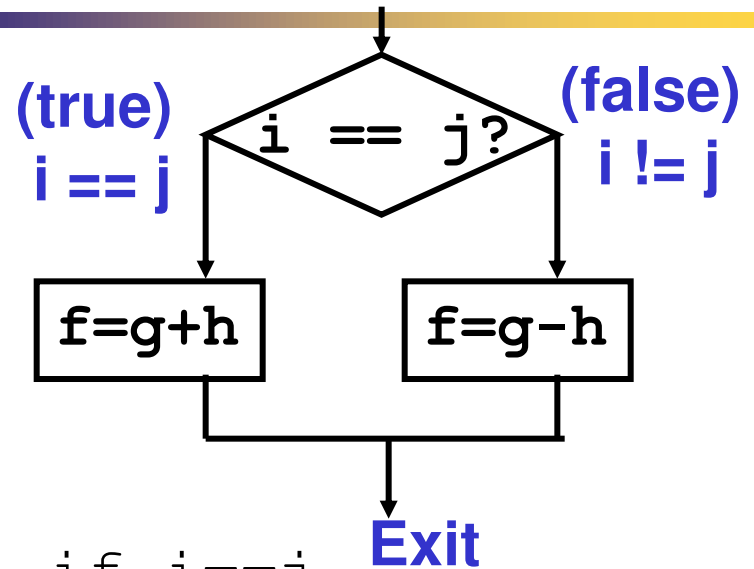

❖Final compiled MIPS code:

```
     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

❖ Simple loop in C;

```
do{
    g--;
    i = i + j;}
while (i != h);
```

❖ Rewrite this as:

```
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

❖Final compiled MIPS code:

```
Loop      SUB r1,r1,#1      ; g--
          ADD r3,r3,r4      ; i=i+j
          CMP r3,r2         ; cmp i,h
          BNE Loop          ; goto Loop
                            ; if i!=h
```

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
```

UCSD

Loops in C/Assembly

❖ There are three types of loops in C:
  - ❖ `while`
  - ❖ `do…while`
  - ❖ `for`

❖ Each can be rewritten as either of the other two, so the method used in the previous example can be applied to `while` and `for` loops as well.

❖ Key Concept: Though there are multiple ways of writing a loop in ARM, conditional branch is key to decision making

Example: The C Switch Statement

❖Choose among four alternatives depending on whether `k` has the value 0, 1, 2 or 3. Compile this C code:

```
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:

```
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: $s0, g: $s1, h: $s2, i: $s3,
 j: $s4, k: $s5
```

Example: The C Switch Statement

```
        CMP  r5,#0           ; compare k, 0
        BNE  L1              ; branch k!=0
        ADD  r0,r3,r4        ; k==0 so f=i+j
        B    Exit            ; end of case so Exit
    L1  CMP  r5,#1           ; compare k, -1
        BNE  L2
        ADD  r0,r1,r2        ; k==1 so f=g+h
        B    Exit            ; end of case so Exit
    L2  CMP  r5,#2           ; compare k, 2
        BNE  L3              ; branch k!=2
        SUB  r0,r1,r2        ; k==2 so f=g-h
        B    Exit            ; end of case so Exit
    L3  CMP  r5,#3           ; compare k, 3
        BNE  Exit            ; branch k!=3
        SUB  r0,r3,r4        ; k==3 so f=i-j
Exit
```

UCSD

Predicated Instructions

❖All instructions can be executed conditionally.
Simply add {EQ,NE,LT,LE,GT,GE, etc.} to end

C source code

ARM instructions

```
if (r0 == 0)
{
  r1 = r1 + 1;
}
else
{
  r2 = r2 + 1;
}
```

unconditional

```
  CMP r0, #0
  BNE else
  ADD r1, r1, #1
  B end
else
  ADD r2, r2, #1
end
  ...
```

- 5 instructions
- 5 words
- 5 or 6 cycles

conditional

```
  CMP r0, #0
  ADDEQ r1, r1, #1
  ADDNE r2, r2, #1
  ...
```

- 3 instructions
- 3 words
- 3 cycles

UCSD

Conclusions

❖ 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
```

    ❖ New:

```
CMP, B{EQ,NE,LT,LE,GT,GE}
```