


BROWN

Computer System Design

Lecture 6: Control Flow Instructions

Prof. R. Iris Bahar
EN164
February 8, 2007

Reading: Appendix B, section B.1-B.7,
Appendix J (optional)



BROWN

Course Website and TA Hours

TA Hours for lab 1:

Week	Mon.	Tues.	Wed.	Thurs.	Fri.	Sat.
2/5 – 2/10		Mickey 3 – 6pm	Brian 7–10pm	Brian 7–10pm		Mickey 1 – 4pm
2/12 – 2/16	Brian 7–10pm	Mickey 3 – 6pm	Mickey 1 – 4pm	Brian 7–10pm		

Please confirm you have an account on the ENGIN cluster ASAP !!

Updates to these hours will also be posted on the course web page

EN164
Lecture 6-2




BROWN

Jobs at Cisco

- Cisco is looking for full-time as well as summer intern hires.
- Jobs for both HW and SW engineers to work on:
 - wireless LAN,
 - IP telephony,
 - security,
 - digital video,
 - network management, routing, switching ...
- For more information, and to apply, go to:
<http://tools.cisco.com/careers/applicant/ciscorm/university/applicant/>

EN164
Lecture 6-3




BROWN

Engineers Week

- February 18-24
- Wednesday, Feb 21, 5pm:
 - Dinner with Cisco and speaker on recycling
 - How does technology make a social impact
- Thursday, Feb 22, 5pm
 - Engineering as art photography show
 - Cash prizes awarded
- Friday, Feb. 23, 6pm
 - Elevator pitch competition on the RI wireless initiative
 - Pitch an idea that uses the proposed statewide network
- Saturday, Feb. 24, 10am
 - Build a functional product from recycled materials

EN164
Lecture 6-4




BROWN

Interpreting Memory Addresses

- Most computers are byte addressed and also allow access to half words (16 bits), words (32), and double words (64)
- Accesses are usually required to be aligned: a half word cannot have an odd address, a double word must have an address A , where $A \bmod 8 = 0$, etc.
- Misalignment increases hardware complexity and worsens performance (if data cross cache line boundaries)

EN164
Lecture 6-6



BROWN

Little and Big Endian

- Consider a 64-bit quantity, composed of bytes 0-7 (LSB-MSB)
- In Little-Endian format, memory address A will contain byte 0, address $A+1$ will contain byte 1,...address $A+7$ will contain byte 7
 - Advantage: easier to organize bytes, half-words, words, double words, etc. into registers (Alpha, x86)
- In Big-Endian format, memory address A will contain byte 7, address $A+1$ will contain byte 6,... address $A+7$ will contain byte 0
 - Advantage: values are stored in the order they are printed out, the sign is available early (Motorola)

EN164
Lecture 6-7

BROWN

“Endianness” Example

- Consider the hexadecimal number:
MSB 0x 43fa27c77156ab91 **LSB**
- Two options:
 - 43fa27c77156ab91 ← **Little-endian**
 - address 7 6 5 4 3 2 1 0
 - 91ab5671c727fa43 ← **Big-endian**

EN164
Lecture 6-8

BROWN

Common Operations

Operator Type	Examples
Arithmetic/Logical	Add, sub, and, or, mult, div
Data transfer	Loads/stores
Control	Branch, jump, call, return
System	OS call, virtual memory management
Floating point	FP add, sub, mult, div
Decimal	Decimal add, sub, mult, decimal to character conversions
String	Move, compare, search
Graphics	Compression/decompression, vertex/pixel ops

EN164
Lecture 6-9

BROWN

Operation Frequency

Top 10 instructions executed for a collection of integer programs

80x86 instruction	Integer average (% total executed)
Load	22%
Conditional branch	20%
Compare	16%
Store	12%
Add	8%
And	6%
Sub	5%
Move register-register	4%
Call/Return	2%

EN164
Lecture 6-10

BROWN

Control Transfer Instructions

- Conditional branches (75% - Integer) (82% - FP)
- Jumps (6% - Integer) (10% - FP)
- Procedure calls/returns (19% - Integer) (8% - FP)
- Design issues:
 - How do you specify the condition?
 - How do you specify the target address?
 - What happens on a procedure call/return?

EN164
Lecture 6-11

BROWN

Specifying the Condition

Name	Examples	How condition is tested	Advantages	Disadvantages
Condition Code (CC)	80x86, ARM, PowerPC, SPARC	Tests special bits set by ALU ops	Sometimes condition is set for free	CC is extra state. Instructions cannot be re-ordered
Condition Register	Alpha, MIPS	Comparison sets register and this is tested	Simple	Register pressure
Compare and branch	PA-RISC, VAX	Comparison is part of the branch	One instruction instead of two	Complex pipelines

EN164
Lecture 6-13

BROWN

Control

- Decision making instructions
 - alter the control flow,
 - i.e., change the "next" instruction to be executed
- MIPS conditional branch instructions:


```
bne $t0, $t1, Label # branch if not equal
beq $t0, $t1, Label # branch if equal
```
- Example (if): if (i==j) h = i + j;


```
bne $s0, $s1, Label
add $s3, $s0, $s1
Label:    . . . .
```

EN164
Lecture 6-14

Control

- MIPS unconditional branch instructions:


```
j label
```
- Example (if - then - else):


```
if (i!=j)          beq $s4, $s5,Label1
    h=i+j;          add $s3, $s4, $s5
else              j Label2
    h=i-j;          Label1: sub $s3, $s4, $s5
                  Label2: ...
```

EN164
Lecture 6-15

Control

- Example (loop):


```
Loop: ---
    i=i+j;
    if(i!=h) go to Loop
    ---
```
- Loop: ---


```
add $s1, $s1, $s2    #i=i+j
bne $s1, $s3, Loop
---
```

EN164
Lecture 6-16

Control Flow

- We have: beq, bne, what about Branch-if-less-than?
- New instruction: set on less than


```
if $s1 < $s2 then
    $t0 = 1
else
    $t0 = 0
```

```
slt $t0, $s1, $s2
```
- slt and bne can be used to implement branch on less than


```
slt $t0, $s1, $s2
bne $t0, $zero, Less
```
- Note that the assembler needs a register to do this.
- We can now build general control structures

EN164
Lecture 6-17

Translating C to Assembly

Assume an accumulator architecture. How do I translate the "while" loop into assembly?

```
//sum the first n-1 integers
int sum=0;
int i=0;
while(i < n){
    sum= sum + i;
    i++;
}
```

```
init:  LOCO 0
      STOD sum
      STOD i
loop:  LODD i
      SUBD n
      JPOS loopend
body:  LODD sum //load sum into accumulator
      ADDD i //add i to it
      STOD sum
incr:  LOCO 1
      ADDD i
      STOD i
      JUMP loop
loopend: ...
```

Through which means is the condition tested here?

EN164
Lecture 6-18

C code to assembly translation

Fragment in SPARC assembly

```
start:
temp = y;
while( temp > 0 ) {
    x = x + z;
    temp = temp - 1;
}
```

```

set    y, %r1          ! use %r2 for temp
ld     [%r1], %r2
set    z, %r1
ld     [%r1], %r3      ! use %r3 for z
mov    %r0, %r4       ! use %r4 for x
add    %r2, 1, %r2     ! set up for decrement
ba     test           ! test the loop condition
top:  add    %r4, %r3, %r4 ! x + z -> x
test: subcc  %r2, 1, %r2 ! temp - 1 -> temp
      bg     top       ! temp > 0 ?
      set    x, %r1
      st     %r4, [%r1] ! store x
```

branch always →

Branch if greater than 0 →

Through which means is the condition tested here?

EN164
Lecture 6-19

Specifying the Target Address

- PC-Relative:** Needs fewer bits to encode, independent of how/where the compiled code is linked, used for branches and jumps — typically, the displacement needs 4-8 bits
- Register-indirect jumps:** The address is not known at compile-time and has to be computed at run-time
 - procedure returns
 - case statements
 - function pointers
 - dynamically shared libraries

EN164
Lecture 6-20

BROWN

Procedure Calls

- Need to maintain a stack of return addresses (in memory or in hardware)
- Can copy and save all registers together or this can be done selectively
- Who is responsible for saving registers?
 - **Caller saving:** correctness issues (global register has to be made available to other procedures), it only saves values that it cares about
 - **Callee saving:** it saves only as many registers as it needs (provided it doesn't call other procedures)
 - A combination of both is typically employed

EN164
Lecture 6-21

BROWN

DSP extensions

- Multiply-accumulate (MAC)
 - Key feature for every digital signal processor (DSP) architecture
 - Used to efficiently implement digital filters
 - Multiplies tend to be on shorter words (e.g., 16 bits)
 - Accumulator tends to be on longer words (e.g., 64 bits).
- You can try out MAC instruction in lab 1
- Modulo Addressing
 - Extra addressing mode helpful for handling circular buffers
 - Also good for making filter algorithms more efficient
- Saturating arithmetic is also typically used in DSP processors.

EN164
Lecture 6-22

BROWN

Multimedia extensions

- Graphics system use 8 bits to represent each of the 3 primary colors plus 8 bits for a location of a pixel
- Audio samples can also be represented with 16 bits
- Microprocessors have special support so bytes and half words take up less space when stored in memory.
- Many graphics and audio applications perform the same operation on vectors of these data.
 - Partition the carry chains within a 64-bit ALU, and perform simultaneous operations on short vectors of eight 8-bit operands, four 16-bit operands, or two 32-bit operands.
 - Call *subword parallelism* or *SIMD*

EN164
Lecture 6-23

BROWN

Instruction Set Encoding

- Operations are easy to encode efficiently – the key issues are the number of operands and their addressing modes
- Few addressing modes → low complexity in decoding and pipelining, but greater code size
- Fixed instruction lengths → low complexity in decoding, but greater code size

EN164
Lecture 6-24