More SASM Instructions

August 30, 2000

CSC201 Section 002
Fall, 2000
SASM I/O "instructions"

- Not really Pentium instructions; actually, macros or procedure calls
  - we'll study real I/O later
- `get_ch A` -- read a character from the keyboard, result is in `A`
- `put_ch A` -- output the byte-sized value at `A` to the console
SASM I/O (cont.)

• put_i A -- output the doubleword-sized integer value at A to the console
  - does formatting (leading zero suppression, minus sign if negative)

• put_fp A -- output the doubleword-sized floating point value at A to the console,
  - with formatting

• put_str A -- output the null-terminated string value starting at A
• data section ---
  aString DB 61h, "bcd", 45h, 0ah, 0dh, 0
  bNum    DD  -231
  cNum    DD  27.62

• code section ---
  put_ch aString
  ; output the character "a" (hex 61) to the screen
  put_i bNum
  ; output the decimal value -231
  put_i aString
  ; output the decimal value 1633837924
     (61626364h)
  put_fp bNum
  ; output who knows what
  put_fp cNum
  ; output 27.62
  put_str aString
  ; output the string "abcdE" followed by
     newline/carriage return
Flow of Control

• Executing statements conditionally, or a variable number of times
  - example: if-then-else, while, for, do, case, ...

• Conditions: the state of the sign flag SF, and the zero flag ZF
  - set by "most recently executed" arithmetic instruction
The Compare Instruction

- Arithmetic instructions generate a result *and* set condition codes
- The "compare" instruction just sets the condition codes
- `compare A,B` -- subtract B from A (doubleword-sized) and set condition flags according to result (but do not store the result)
- `compareb A,B` -- same, but A and B are byte-sized
Branch Instructions

• Changes what the next instruction that gets executed will be
  – updates the "special register" in the instruction cycle!

• Unconditional branch: br label
  – like "goto" in a high-level language

• Watch out for "spaghetti code"!
Example

Main:

move a,b
br skip_over
add a,c

Skip_over:

...
Conditional Branch Instructions

- There are 6 conditions involving comparison with zero
- `blz lab1` -- branch if the result was negative to `lab1`

<table>
<thead>
<tr>
<th>C operator</th>
<th>SASM instruction</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>blz</td>
<td>negative</td>
</tr>
<tr>
<td>&lt;=</td>
<td>blez</td>
<td>negative or zero</td>
</tr>
<tr>
<td>=</td>
<td>bez</td>
<td>zero</td>
</tr>
<tr>
<td>!=</td>
<td>bnz</td>
<td>non-zero</td>
</tr>
<tr>
<td>&gt;=</td>
<td>bgez</td>
<td>positive</td>
</tr>
<tr>
<td>&gt;</td>
<td>bgz</td>
<td>positive (not zero)</td>
</tr>
</tbody>
</table>
The If Statement With Simple Conditions

• "simple conditions" means no ANDs, ORs, XORs, NOTs

• if condition is true, execute statement
  - Otherwise, skip it (branch over it)!
Example

• C:

```c
if (a == b)
    c = 1;
```

• SASM:

```assembly
compare a,b
bnz skip_over
move c, 1
skip_over: ...
```
Another Example

• C:

```c
if (d !> e)
    f = 2;
```

• SASM:

```assembly
compare d, e
bgz skip_over
move f, 2
skip_over:    ...
```
Compound Conditions: OR

• C:

```c
if ((a == b) || (c == d))
    e = 3;
```

• SASM:

```sasm
compare a,b
bez justdoit
compare c, d
bez justdoit
br skip_over
justdoit:
    move e,3
skip_over: ...
```
Compound Conditions: AND

- **C:**
  ```c
  if ((w > x) && (y == 2))
      z = z + 3;
  ```

- **SASM:**
  ```assembly
  compare w,x
  blez skip_over
  compare y,2
  bnz skip_over
  justdoit:
      iadd z,3
  skip_over: ...
  ```
You Try It
If-Then-Else

- **C:**

```c
if (a < b)
    c = 3;
else
    c = 4;
```

- **SASM:**

```assembly
compare a,b
blz ifpart
elsepart:
    move c,4
br carry_on
ifpart:
    move c,3
carry_on: ...
```