Control Flow

CSE 132

Logistics

• Assignment 12 due date moved to April 27
  – Only one input set
  – In fixedPoint.ino, remove the “unsigned” type
  – Adding extra credit portion (not yet posted)
• Studio today
  – Flow control topics
  – Material is fair game for final exam
• Last lecture and studio will be review for final
  – Material will be cumulative
  – Tuesday, May 10, 10:30am-12:30pm, Lab Sci 300

Logistics (cont.)

• Late tickets
  – Even if you have consumed all of your late tickets, turn in (get checked out on) all the assignments
  – Partial credit is available, even if late
• Finishing assignments
  – All assignments must be checked out by 5:30pm on Wed., April 27

Integer Multiplication

\[
\begin{align*}
xxxxxxx & \times yyyy = zzzzzzzzzzzzzzzzzzzzzz \\
\text{or} & \\
\end{align*}
\]

Q15 Multiplication

\[
\begin{align*}
x.xxxxxxxx & \times y.yyyyyyyyyyyyyy = z.zzzzzzzzzzzzzzzzzzzzzzzzzzzzzz \\
\text{or} &
\end{align*}
\]

Assembly Control Flow

• Unconditional Jump –
  \[ \text{jmp [label]} \]
  e.g.,
  \[ \text{jmp L1} \]
  \[ \text{L1: target instruction} \]
  \[ \text{ijmp } \text{indirect, dest in Z} \]
Conditional Control Flow

- In AVR, separate expression eval and cond branch inst.
- Compare –
  \[ cp \quad Rd, Rr \]
- Perform operation \( temp = Rd - Rr \), throw away temp and set flags in \( SREG \) based on results of subtraction
- Flags can also be set as a result of normal arithmetic and/or logical operations

Conditional Jumps

\[ br[cond] \quad [label] \]
e.g.,
\[ brne \quad j\_loop \]

- There are three classes of conditionals:
  - General (Simple)
  - Unsigned
  - Signed

General Conditionals

- \( breq \) zero (\( Z \) set)
- \( brne \) not zero (\( Z \) clear)
- \( brcs \) carry (\( C \) set)
- \( brcc \) no carry (\( C \) clear)

Signed Conditionals

- \( brge \) greater than or equal (\( Rd \geq Rr \))
- \( brlt \) less than (\( Rd < Rr \))

Unsigned Conditionals

- \( brsh \) same or higher (\( Rd \geq Rr \))
- \( brlo \) lower than (\( Rd < Rr \))

Control Flow in C

if ... then ...
e.g.,
if \( ([cond expr]) \) {
  \[ \text{var1} = \text{var1} + \text{var2}; \]
  \[ \text{var2} = 0; \]
} 
...
if ... then
if (var1 > var2) {
    var1 = var1 + var2;
    var2 = 0;
}
...

if ... then else
if (var1 == var2) {
    var1 = var1 + var2;
    var2 = 0;
} else {
    var2 = var2 + var1;
    var1 = var2;
}
...

Conditional if ... then else
if (((cond1) && (cond2)) || (cond3)) {
    [true body]
} else {
    [false body]
}

Evaluation order for above compound expression:

Conditional if ... then else
if (((cond1) && (cond2)) || (cond3))

• Note: evaluation order of compound expression is left to right, only conditions that need to be evaluated are evaluated
for loop

for ([ind var] = [init val]; [cond expr]; [update ind var]) {
  [loop body]
}
[main body]

e.g.,
for (i=0; i<24; i++) {
  mask = 1 << i;
  status_bit[i] = status & mask;
  status_bit[i] >>= i;
}

while loop

while ([cond expr]) {
  [loop body]
}[main body]

• Assembly
  while_loop:
  cp [cond expr oper]
  br[cond] exit_loop
  [loop body]
  jmp while_loop
exi

for loop

for ([ind var] = [init val]; [cond expr]; [update ind var]) {
  [loop body]
}
[main body]

• Assembly
  for_loop: ldi [ind var], [init val]
  for_loop: cp [cond expr]
  br[!cond] loop_exit
  [loop body]
  [update ind var]
  jmp for_loop
loop_exit:
[main_body]