Translation of Branch Statement

IF-ELSE statement:

Take an example of this C code. This code is inside some function.

```
int a = 4;
int b = 8;
int d = 0;
if( a > b)
{
   d = 1;
}
else
{
   d = 2;
}
```

Generated assembly code:

```
   movl   $4, -4(%ebp)
   movl   $8, -8(%ebp)
   movl   $0, -12(%ebp)
   movl   -4(%ebp), %eax
   cmpl   -8(%ebp), %eax
   jle     .L2
   movl   $1, -12(%ebp)
   jmp     .L3
.L2:
   movl   $2, -12(%ebp)
.L3:
```

Location of local variables of the stack (local variables are explained here (memorymanagement.html))

```
a => -4(%ebp)
b => -8(%ebp)
d => -12(%ebp)
```

The use of registers as temporary memory is described here (arithmeticop.html#tempVaribaleUsage)

Comments on generated code:

```
# a = 4
   movl    $4, -4(%ebp)

# b = 8
   movl    $8, -8(%ebp)

# d = 0
   movl    $0, -12(%ebp)

# tmp = a
   movl    -4(%ebp), %eax

# compare b and tmp
   cmpl    -8(%ebp), %eax

# jump to label .L2 (i.e. to else block of original C code) if
# tmp is less than or equals to b
   jle .L2

# Code of if block starts now
# d = 1
   movl    $1, -12(%ebp)

# last instruction for the if block.
# jump to the instruction after else block
   jmp  .L3

# Code of else block starts now at label .L2
.L2:

# d = 2
   movl    $2, -12(%ebp)

# else block has finished.
# This is just after the else block
.L3:
```

Switch-Case Statement

Take an example of this C code. This code is inside some function.

```
int a = 4;
int b = 8;
int d = 0;
switch(b)
{
   case 2:
      a++;
      break;
   case 8:
      b++;
      break;
   default:
      d++;
      break;
}
```

Generated assembly code:

```
   movl    $4, -4(%ebp)
   movl    $8, -8(%ebp)
   movl    $0, -12(%ebp)
   movl    -8(%ebp), %eax
   cmpl    $2, %eax
   je .L3
   cmpl    $8, %eax
   je .L4
   jmp      .L7
.L3:
   addl    $1, -4(%ebp)
   jmp      .L5
.L4:
   addl    $1, -8(%ebp)
   jmp      .L5
.L7:
   addl    $1, -12(%ebp)
.L5:
```

Location of local variables of the stack (local variables are explained here (memorymanagement.html))

```
a => -4(%ebp)
b => -8(%ebp)
d => -12(%ebp)
```

The use of registers as temporary memory is described here (arithmeticop.html#tempVaribaleUsage)
Comments on generated assembly code:

```
# a = 4
# b = 8
# d = 0
   movl    $4, -4(%ebp)
   movl    $8, -8(%ebp)
   movl    $0, -12(%ebp)

# tmp = b
   movl    -8(%ebp), %eax

# compare tmp to 2
   cmpl    $2, %eax

# if above comparison succeeds then goto .L3. At .L3 there is code of first
# case of the switch block
   je .L3

# compare tmp to 8
   cmpl    $8, %eax

# if above comparison succeeds then goto .L4. At .L4 there is code of
# second case of the switch block
   je .L4

# this is the default case. Jump to the label where the code of default
# case is generated.
   jmp      .L7

# code for the fist case of the switch
.L3:
   addl    $1, -4(%ebp)
   jmp      .L5

# code for the 2nd case of the the switch
.L4:
   addl    $1, -8(%ebp)
   jmp      .L5
```

```
# code for the default case of the switch
.L7:
    addl    $1, -12(%ebp)

# label just after the switch-case block
.L5:
```

At the end of the code of each case block (except default) there is a jump statement to the end of the switch-case. This jump is generated because you wrote a break statement in the case block. If you do not write the break then this jump will not be generated and code for the next case block will be executed. This is in accordance with C specification.

‹ Translation of Bitwise Operations (/cin/bitwiseop.html)
up (/cin/cin.html)
Translation of Loop › (/cin/loop.html)
