50.002 COMPUTATIONAL STRUCTURES

INFORMATION SYSTEMS TECHNOLOGY AND DESIGN

Problem Set 9

1 GCD

Consider the following implementation of an algorithm for finding the greatest common divisor of two integers:

```
int gcd(int a,int b) {
 if (a == b) return a;
 if (a > b) return gcd(a-b,b);
 return gcd(a,b-a);
}
```

The C compiler has compiled this procedure into the following code for an unpipelined Beta processor:

```
gcd:
PUSH (LP)
PUSH (BP)
MOVE (SP, BP)
PUSH (R1)
PUSH (R2)
LD (BP, -12, R0)
LD (BP, -16, R1)
CMPEQ (R0, R1, R2)
BT (R2, L1)
CMPLE (R0, R1, R2)
BT (R2, L2)
PUSH (R1)
SUB (R0, R1, R2)
PUSH (R2)
BR (gcd, LP)
DEALLOCATE (2)
BR (L1)
L2:
```

```
SUB (R1, R0, R2)
PUSH (R2)
PUSH (R0)
BR (gcd, LP)
DEALLOCATE (2)
L1:
POP (R2)
POP (R1)
MOVE (BP, SP)
POP (BP)
POP (LP)
JMP (LP)
```

Answer the following questions: **For convenience sake, although all data is in 32-bit format, for this answer, we omit the leading zeroes**.

1. The program above contains the instruction LD(BP,-16,R1). Explain what the compiler was trying to do when in generated this instruction.

   Solution:
   Recall that BP - 12 **is always the first argument of the function**. Hence one word / one line above (BP - 12), i.e: BP - 16 is the second argument of the function. The compiler is loading the value of the second argument (”b”) so it could compute ”a == b”.

2. What are the contents of the memory location holding the instruction BR(L1)?

   Solution:
   ’L1’ is a label. **Recall that to quickly compute ’literal’ you can count the number of instruction lines between BR (the instruction that uses ’label’) and the ’label’ instruction, and subtract it by 1, and convert it to 16 bits**:

   (a) The location of L1, from the assembly code above, is located 1 (from SUB(R1, R0,R2) + 2 (from PUSH(R2)) + 2 (from PUSH(R0)) + 1 (from BR(gcd, LP)) + 1 (from DEALLOCATE(2)) + 1 (from POP(R2), which is the location of L1)= 8 lines (8 words) away from the instruction BR(L1).

   (b) Therefore the 16-bit ’literal’ value for the 32-bit instruction of BR(L1) is 8 - 1 = 7 : 0000 0000 0000 0111.

   (c) The OPCODE for BR : BEQ(R31, L1, R31) is 011101. Ra is R31: 111111, and Rc is also R31: 111111.

   (d) Hence the 32-bit instruction of BR(L1) is : 011101 11111 11111 0000 0000 0000 0111.

   In hex, BR(L1) is encoded as 0x77FF 0007. Remember that PUSH is actually a macro that expands into two Beta instructions, so there are 7 instructions between L1 and L2.

3. When the instruction labeled ”L1” is executed, what is the best characterization of the contents of R0?

   Solution:
   Recall in the 'procedures' notes, that R0 is a reserved register for return value. Since L1 is just at the beginning of the procedure return sequence, so at that point R0 should contain the value to be returned to the caller.

4. Looking at the code, a student suggests that both DEALLOCATE instructions could be eliminated since deallocation is performed implicitly by the MOVE(BP,SP) instruction in the exit sequence. After calling gcd, would it be possible to tell if the DEALLOCATE instructions had been removed?

   Solution:
   Yes, it would be possible to tell: even though the stack is reset correctly by the MOVE(BP,SP) instruction, that happens after the POP(R1) and POP(R2) instructions, so they will not restore the values of R1 and R2 from the values pushed onto the stack during the entry sequence.

5. How many words of stack are needed to execute gcd(24,16)? Don't forget to include the stack space occupied by the arguments in the initial call.

   Solution:
   gcd(24,16) ≥ gcd(8,16) ≥ gcd(8,8), at which point no more recursive calls are made. So three nested procedure calls are executed, each with a 6-word stack frame (2 args, LP, BP, saved R1, saved R2), for a total of 18 words of stack space.

6. During execution of gcd(28,70), the Beta processor is halted and the contents of portions of the stack are found to contain the following:

```
            ????
        0x00000594
        0x00001234
        0x00000046
        0x0000002A
        0x0000000E
        0x0000001C
        0x00000594
        0x0000124C
BP-->   0x0000002A
        0x0000000E
SP-->   0x00001254
        0x0000000E
```

Figure 1

   What is the value of the second argument (”b”) to the current call to gcd?

   Solution:
   The second argument is located at Mem[Reg[BP] - 16] = 0xE.

7. What is the value in the BP register at the time the stack snapshot was taken?

   Solution:
   0x1264 = (old BP stored in Mem[Reg[BP] - 4]) + (size of one stack frame) = 0x124C + (6 words = 24 bytes = 0x18 bytes) = 0x124C + 0x18

8. What is the correct value for "???" above?

   Solution:
   gcd(28,70) ≥ gcd(28,42) ≥ gcd(28,14), at which point the snapshot was taken. So "???" is the first arguments to the preceding call, i.e., 28.

9. What is the address of the POP(R2) instruction?

   Solution:
   All calls in the execution of gcd(28,70) have a < b, so all the recursive calls are from the second BR(gcd,LP). Using the value of LP stored at Mem[Reg[BP] - 8] which points at the DEALLOCATE instruction, the POP(R2) is stored at location 0x594 + 4 = 0x598.

10. At the time the stack snapshot was taken, what is the significance of the value 0x1254 in the location at <SP>?

    Solution:
    Since SP points to the first unused location on the stack, the value in the location SP points to has no significance to the calculation, it's just ununsed stack space.

11. The stack snapshot was taken just after the execution of a particular instruction. Could the snapshot have been taken just after the execution of the PUSH(R1) instruction near the beginning of the gcd procedure?

    Solution:
    No. Given the values of BP and SP, the PUSH(R2) instruction must have already been executed.