Name ***_Answers_***

CE-1921-11 – Dr. Durant – Quiz 3
Spring 2016, Week 3

1. (7 points) Translate the following C/Java-like function into ARMv4 assembly. Use the standard ARM registers for arguments and return value

```
int logic(int x, int y) {
      if(x>y)
            return 2*x;
      else if (x<y)
            return 3*y;
      else
            return x & y; // bitwise AND
}
```

Answer:

```
logic:      cmp r0,r1 ; x,y
            addgt r0,r0,r0 ; gt = signed >, 2*x = x+x, + often faster
            movlt r2,#3 ; mul doesn't take immediate operand
            mullt r0,r1,r2
            andeq r0,r0,r1
            mov pc,lr

logic2: ; without conditional execution (except on branches)
            cmp r0,r1
            ble L0 ; le = !gt = not >
            add r0,r0,r0 ; if we got here, gt must be true
            b done2
L0:         beq L1
            mov r2,#3
            mul r0,r1,r2 ; if we got here, le & !eq --> lt or <
            b done2
L1:         and r0,r0,r1
done2:      mov pc,lr
```

2. (3 points) Write a main routine that calls your function with the arguments 17 and 33, moves the result to r7, and then hangs/spins on one instruction forever.

Answer:

```
main: mov r0,#17
      mov r1,#33
      bl remainder
      mov r7,r0

end2: b end2
```

Name ***_Answers_***

CE-1921-12 – Dr. Durant – Quiz 3
Spring 2016, Week 3

1. (7 points) Translate the following C/Java-like function into ARMv4 assembly. Use the standard ARM registers for arguments and return value

```
int remainder(unsigned int x, unsigned int y) {
      while(x>y) {
            x -= y;
      }
      return x;
}
```

Answer:

```
remainder: ; with conditional execution
            cmp r0,r1; x,y
            subhi r0,r0,r1 ; hi = unsigned > (not gt, which is signed)
            bhi remainder
            mov pc,lr

remainder2: ; without conditional execution
            cmp r0,r1
            bls done
            sub r0,r0,r1
            b remainder2
done:       mov pc,lr
```

2. (3 points) Write a main routine that calls your function with the arguments 17 and 33, moves the result to r7, and then hangs/spins on one instruction forever.

Answer:

```
main: mov r0,#17
      mov r1,#33
      bl remainder
      mov r7,r0

end2: b end2
```