

CS3021/3421 Tutorial 2

Consider the following C/C++ code segment:

```

_int64 g = 4;

_int64 min(_int64 a, _int64 b, _int64 c) {
    _int64 v = a;
    if (b < v)
        v = b;
    if (c < v)
        v = c;
    return v;
}

_int64 p(_int64 i, _int64 j, _int64 k, _int64 l) {
    return min(min(g, i, j), k, l);
}

_int64 gcd(_int64 a, _int64 b) {
    if (b == 0) {
        return a;
    } else {
        return gcd(b, a % b);
    }
}

_int64 q(_int64 a, _int64 b, _int64 c, _int64 d, _int64 e) {
    _int64 sum = a + b + c + d + e;
    printf("a = %l64d b = %l64d c = %l64d d = %l64d e = %l64d sum = %l64d\n", a, b, c, d, e, sum);
    return sum;
}

```

- Q1. Translate the code segment above into x64 assembly language using the basic code generation strategy outlined in lectures. The % operation can be implemented using the x64 cqo and idiv instructions. Assume you must supply shadow space for the calls to function min in p.
- Q2. Draw a diagram showing the state of the stack at its maximum depth during the calculation of gcd(14, 21).
- Q3. Using Visual Studio (or similar), create an x64 console application with files t2.h and t2.asm containing the x64 assembly language for min, p, gcd and q. Use [t2Test.cpp](#) to test min, p, gcd and q. Hand in code listings for t2.h and t2.asm and screen snapshots showing that your program builds and evidence that it works.
- Q4. Write a function qns() which simply calls printf("qns\n") with and without allocating shadow space. Determine what happens when qns() is executed with and without shadow space (provide a screen snapshot).