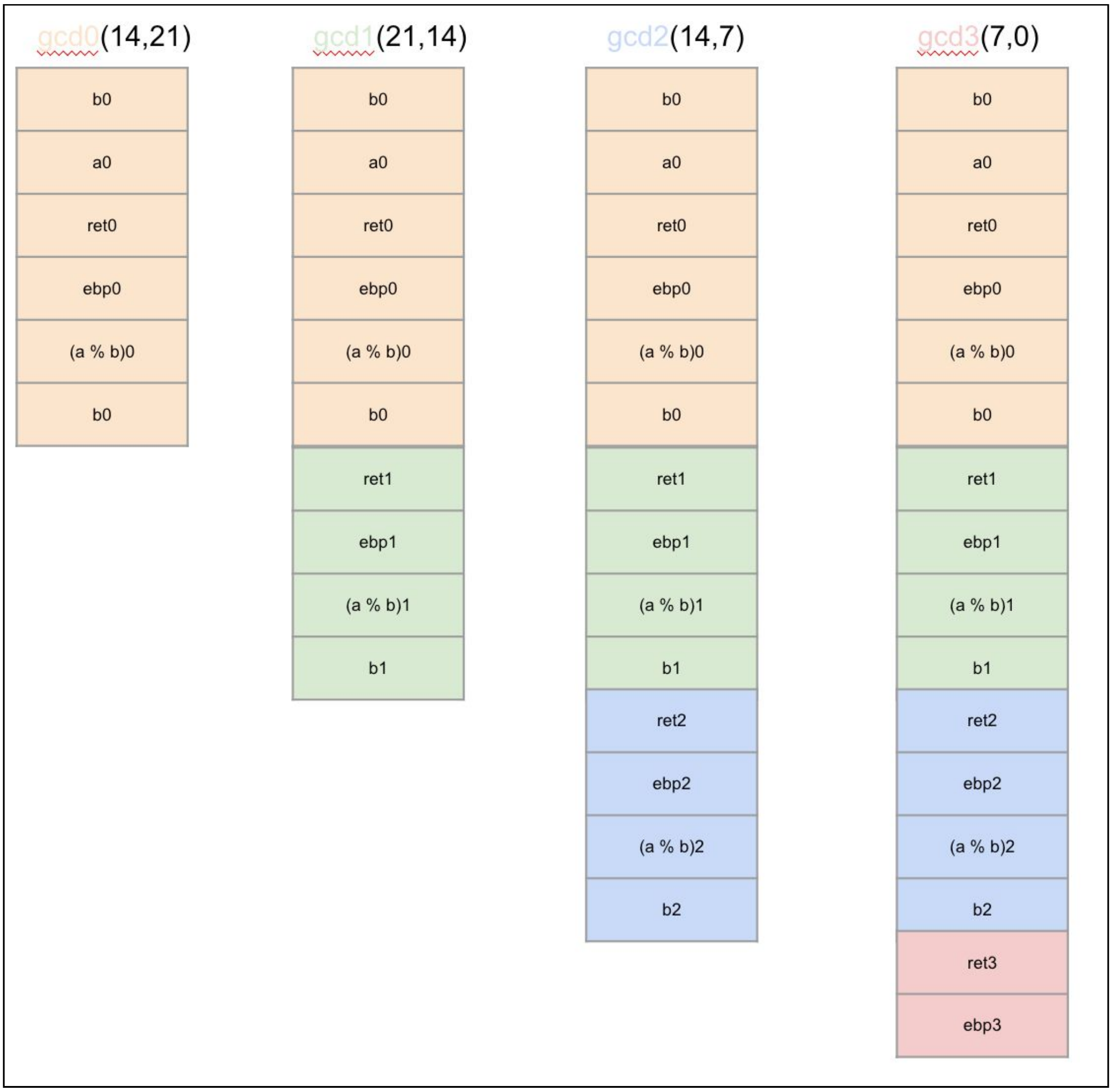
<u>**CS3021 Computer Architecture II - Tutorial 1**</u>

*Student # 16327446 - Brandon Dooley*

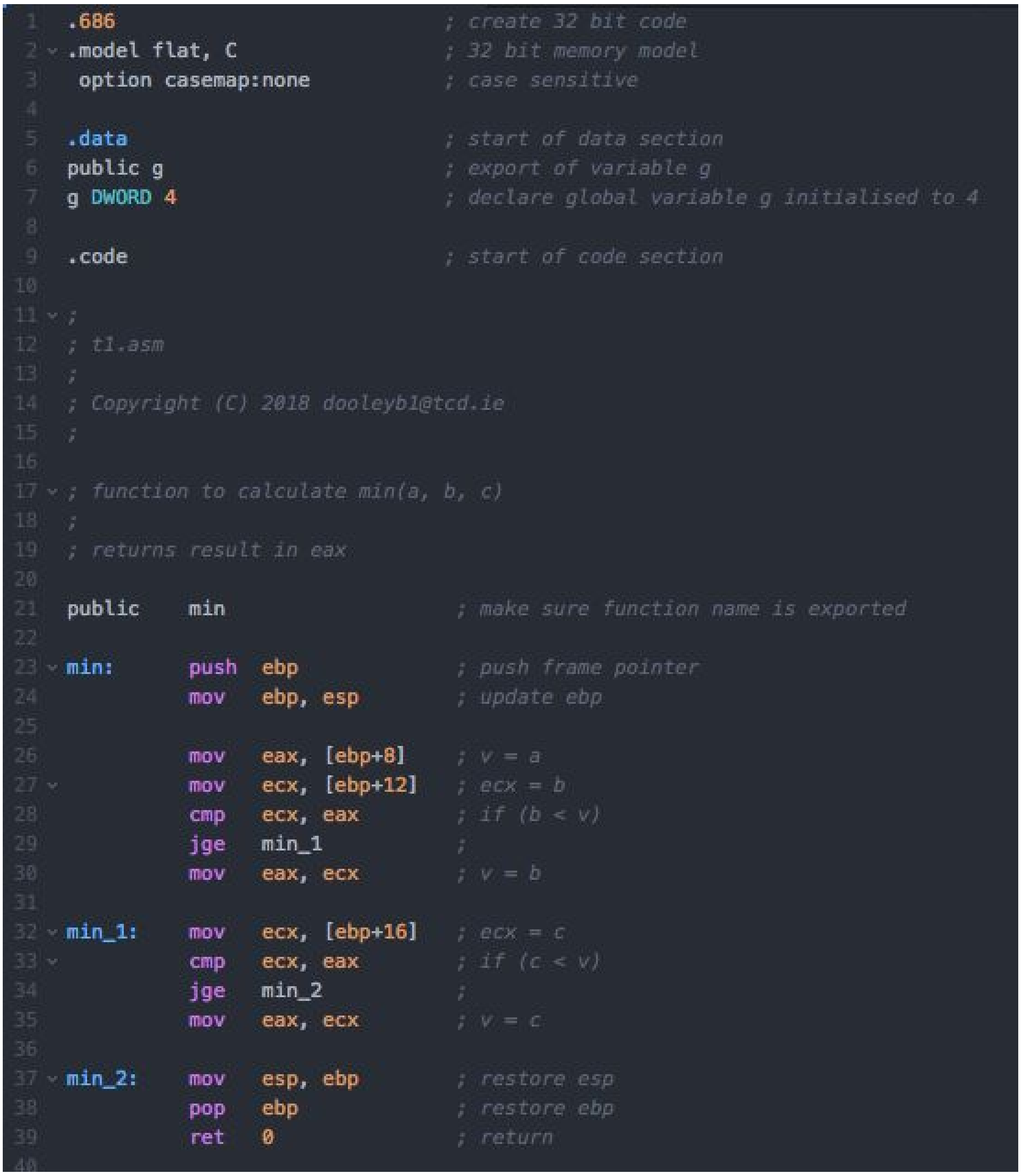
2)

**Maximum Stack Depth (in Frames) =** 4 (See diagram below)



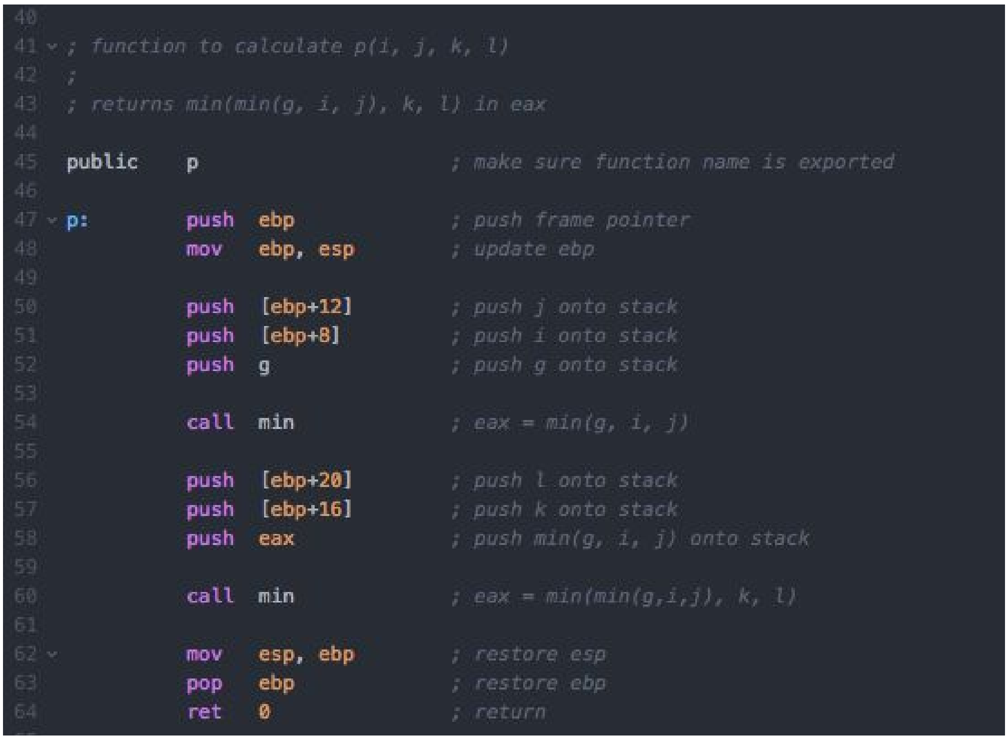
3) Code

**[t1.asm]** - min



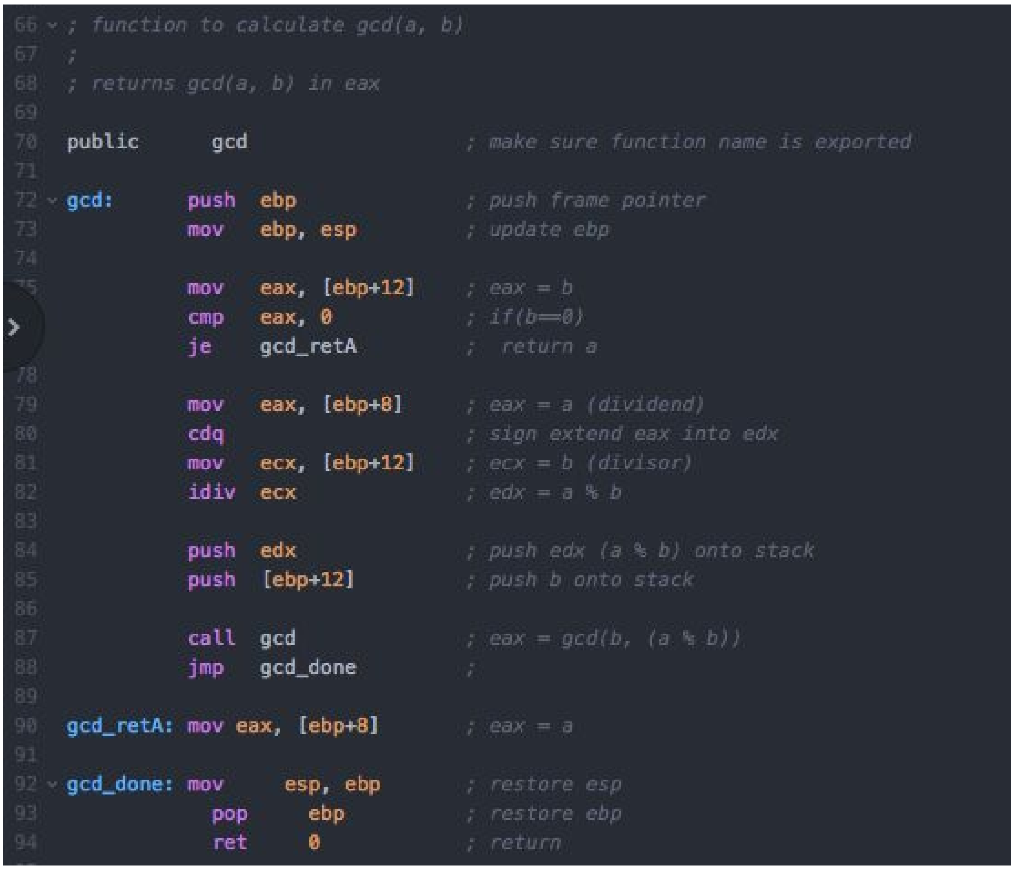
```
.686                           ; create 32 bit code
.model flat, C                 ; 32 bit memory model
 option casemap:none           ; case sensitive

.data                          ; start of data section
public g                       ; export of variable g
g DWORD 4                      ; declare global variable g initialised to 4

.code                          ; start of code section

;
; t1.asm
;
; Copyright (C) 2018 dooleyb1@tcd.ie
;

; function to calculate min(a, b, c)
;
; returns result in eax

public    min                   ; make sure function name is exported

min:      push  ebp             ; push frame pointer
          mov   ebp, esp        ; update ebp

          mov   eax, [ebp+8]    ; v = a
          mov   ecx, [ebp+12]   ; ecx = b
          cmp   ecx, eax        ; if (b < v)
          jge   min_1           ;
          mov   eax, ecx        ; v = b

min_1:    mov   ecx, [ebp+16]   ; ecx = c
          cmp   ecx, eax        ; if (c < v)
          jge   min_2           ;
          mov   eax, ecx        ; v = c

min_2:    mov   esp, ebp        ; restore esp
          pop   ebp             ; restore ebp
          ret   0               ; return
```

[t1.asm] - p



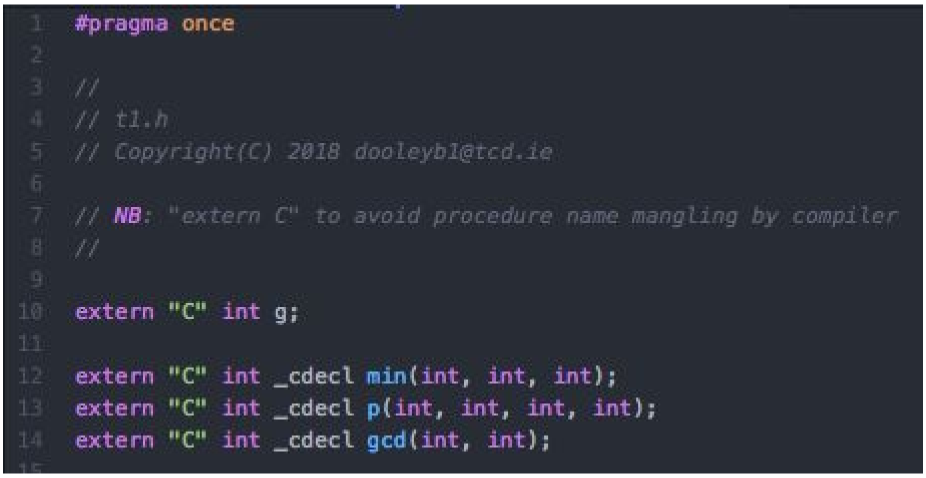
```
; function to calculate p(i, j, k, l)
;
; returns min(min(g, i, j), k, l) in eax

public    p                 ; make sure function name is exported

p:        push  ebp         ; push frame pointer
          mov   ebp, esp    ; update ebp

          push  [ebp+12]    ; push j onto stack
          push  [ebp+8]     ; push i onto stack
          push  g           ; push g onto stack

          call  min         ; eax = min(g, i, j)

          push  [ebp+20]    ; push l onto stack
          push  [ebp+16]    ; push k onto stack
          push  eax         ; push min(g, i, j) onto stack

          call  min         ; eax = min(min(g,i,j), k, l)

          mov   esp, ebp    ; restore esp
          pop   ebp         ; restore ebp
          ret   0           ; return
```

[t1.asm] - gcd



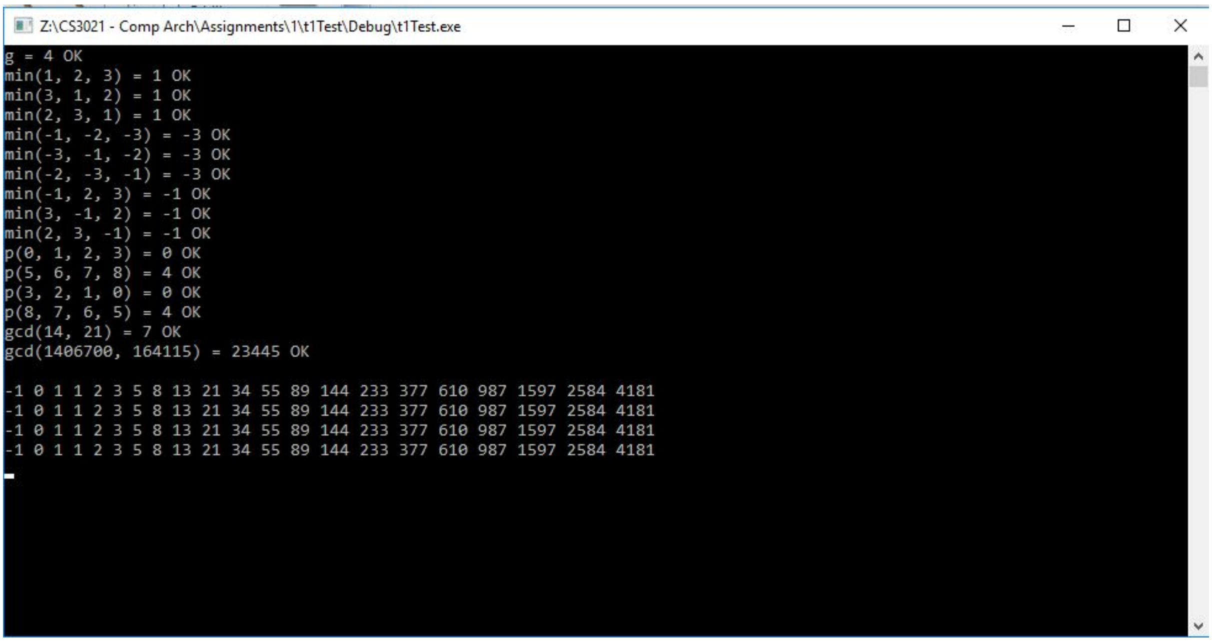
```
; function to calculate gcd(a, b)
;
; returns gcd(a, b) in eax

public      gcd                   ; make sure function name is exported

gcd:        push  ebp             ; push frame pointer
            mov   ebp, esp        ; update ebp

            mov   eax, [ebp+12]   ; eax = b
            cmp   eax, 0          ; if(b==0)
            je    gcd_retA        ;  return a

            mov   eax, [ebp+8]    ; eax = a (dividend)
            cdq                   ; sign extend eax into edx
            mov   ecx, [ebp+12]   ; ecx = b (divisor)
            idiv  ecx             ; edx = a % b

            push  edx             ; push edx (a % b) onto stack
            push  [ebp+12]        ; push b onto stack

            call  gcd             ; eax = gcd(b, (a % b))
            jmp   gcd_done        ;

gcd_retA: mov eax, [ebp+8]        ; eax = a

gcd_done: mov     esp, ebp        ; restore esp
            pop     ebp           ; restore ebp
            ret     0             ; return
```

**[t1.h]**



```
#pragma once

//
// t1.h
// Copyright(C) 2018 dooleyb1@tcd.ie

// NB: "extern C" to avoid procedure name mangling by compiler
//

extern "C" int g;

extern "C" int _cdecl min(int, int, int);
extern "C" int _cdecl p(int, int, int, int);
extern "C" int _cdecl gcd(int, int);
```

3) Console Window



```
g = 4 OK
min(1, 2, 3) = 1 OK
min(3, 1, 2) = 1 OK
min(2, 3, 1) = 1 OK
min(-1, -2, -3) = -3 OK
min(-3, -1, -2) = -3 OK
min(-2, -3, -1) = -3 OK
min(-1, 2, 3) = -1 OK
min(3, -1, 2) = -1 OK
min(2, 3, -1) = -1 OK
p(0, 1, 2, 3) = 0 OK
p(5, 6, 7, 8) = 4 OK
p(3, 2, 1, 0) = 0 OK
p(8, 7, 6, 5) = 4 OK
gcd(14, 21) = 7 OK
gcd(1406700, 164115) = 23445 OK

-1 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
-1 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
-1 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
-1 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
```