<u>CS3021 Computer Architecture II - Tutorial 2</u>

Student # 16327446 - Brandon Dooley

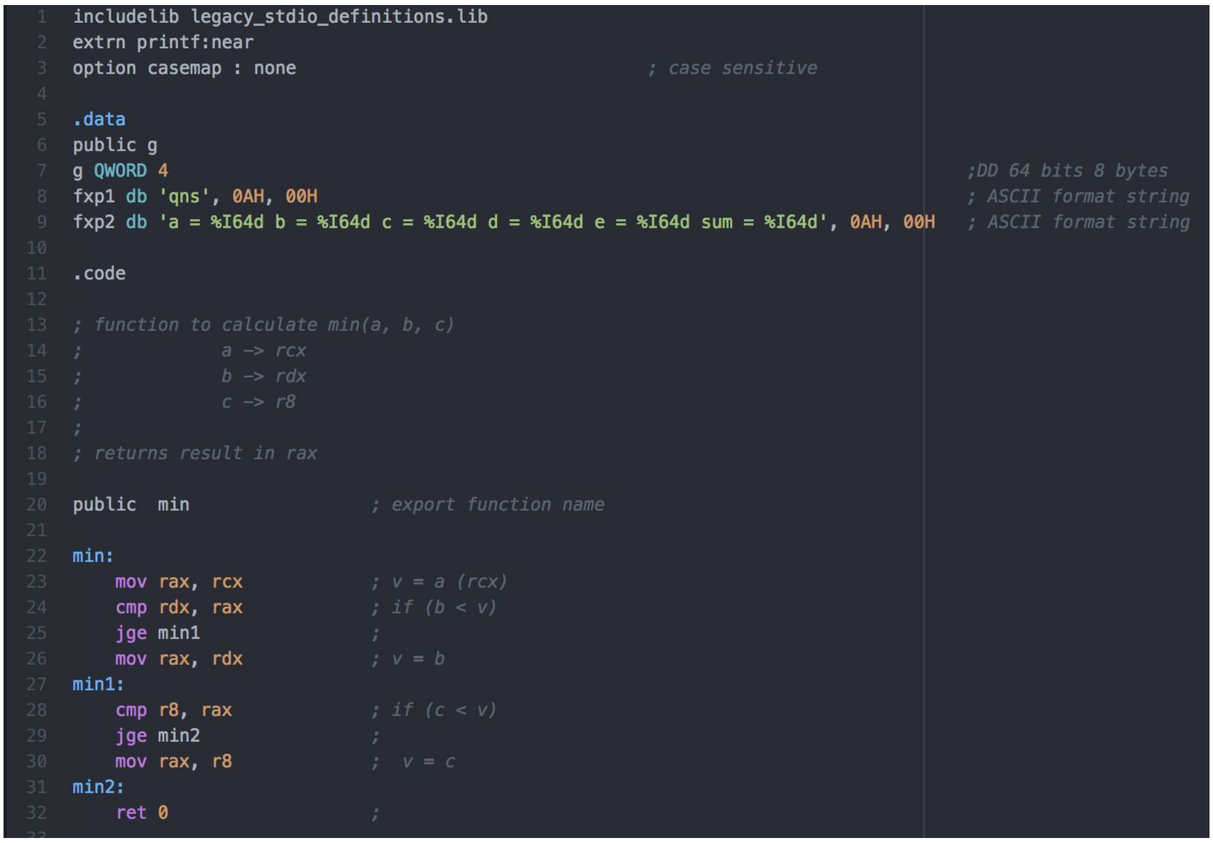
1) As a prerequisite, my code does work, build and run correctly

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

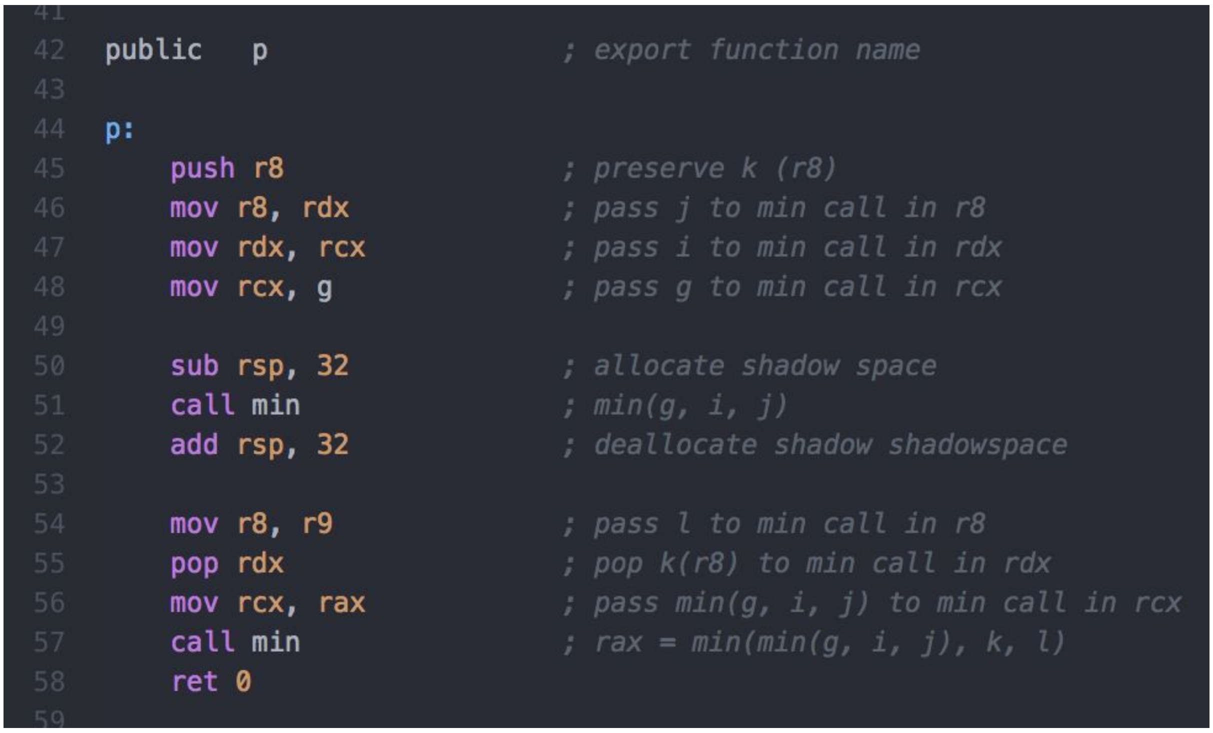
r9
r8
rdx = [b->21]
rcx = [a->14]
ret
r9
r8
rdx = [b->14]
rcx = [a->21]
ret
r9
r8
rdx = [b->7]
rcx = [a->14]
ret
r9
r8
rdx = [b->0]
rcx = [a->7]
ret

3) Code

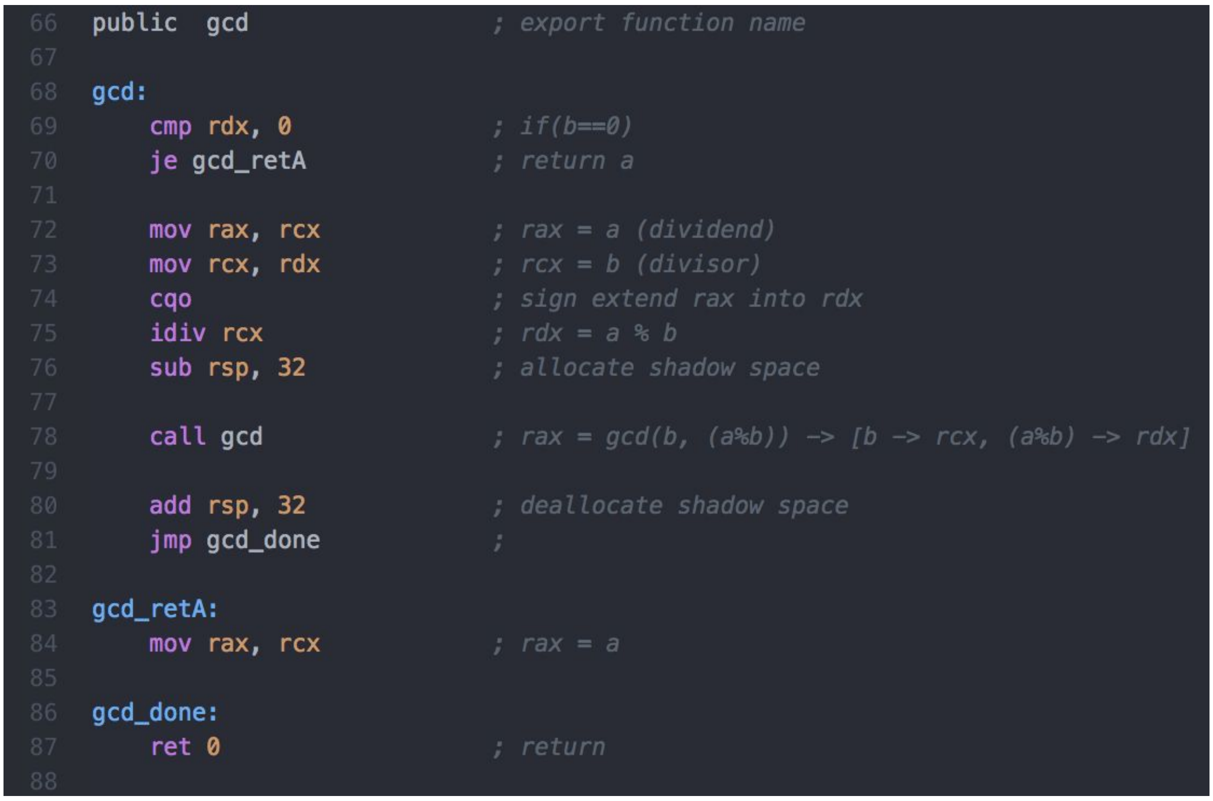
[t2.asm] - min


```
includelib legacy_stdio_definitions.lib
extrn printf:near
option casemap : none                              ; case sensitive

.data
public g
g QWORD 4                                                                          ;DD 64 bits 8 bytes
fxp1 db 'qns', 0AH, 00H                                                            ; ASCII format string
fxp2 db 'a = %I64d b = %I64d c = %I64d d = %I64d e = %I64d sum = %I64d', 0AH, 00H   ; ASCII format string

.code

; function to calculate min(a, b, c)
;              a -> rcx
;              b -> rdx
;              c -> r8
;
; returns result in rax

public  min                     ; export function name

min:
    mov rax, rcx                ; v = a (rcx)
    cmp rdx, rax                ; if (b < v)
    jge min1                    ;
    mov rax, rdx                ; v = b
min1:
    cmp r8, rax                 ; if (c < v)
    jge min2                    ;
    mov rax, r8                 ;  v = c
min2:
    ret 0                       ;
```

[t2.asm] - p

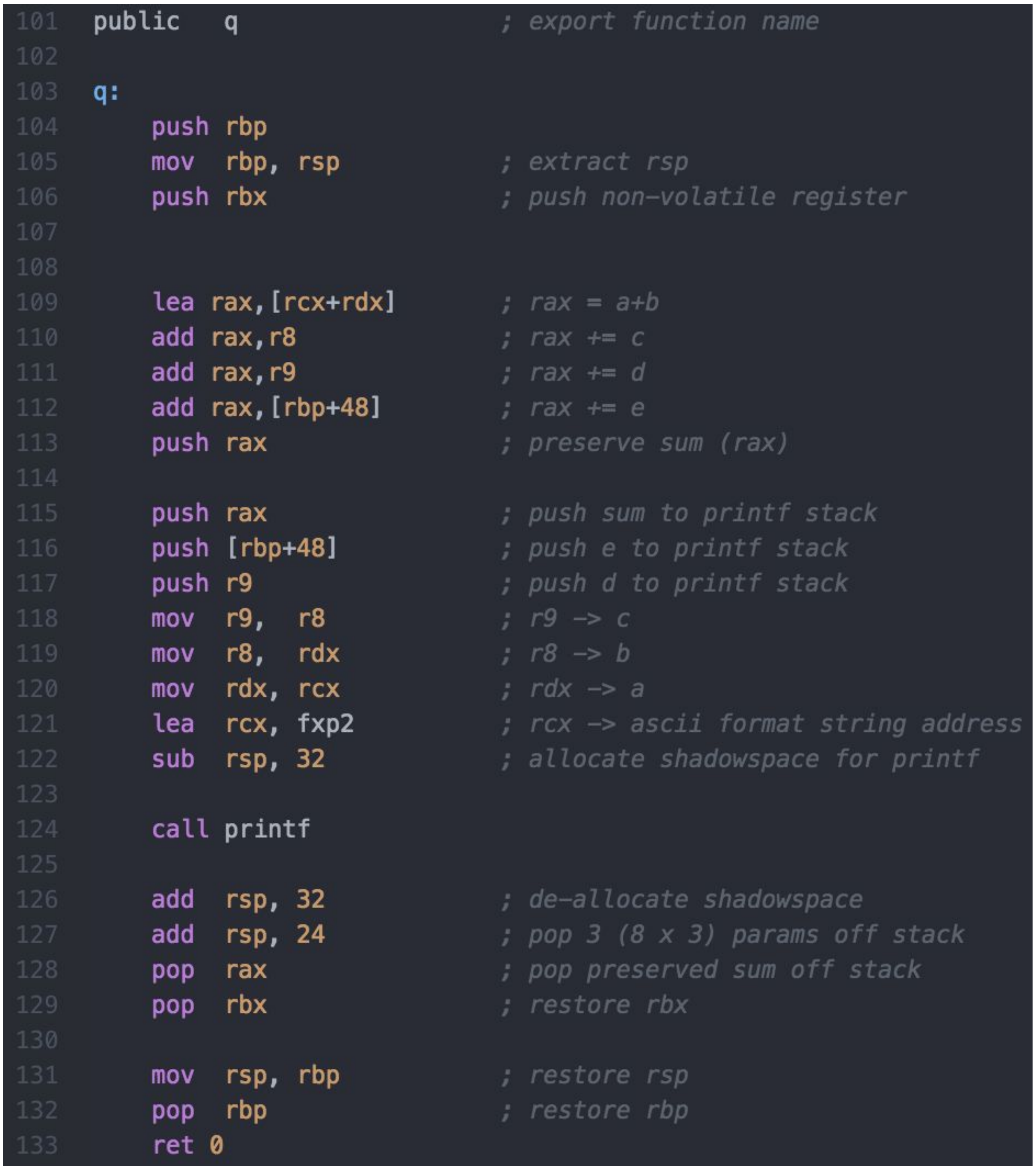

```
public   p                   ; export function name

p:
    push r8                  ; preserve k (r8)
    mov r8, rdx              ; pass j to min call in r8
    mov rdx, rcx             ; pass i to min call in rdx
    mov rcx, g               ; pass g to min call in rcx

    sub rsp, 32              ; allocate shadow space
    call min                 ; min(g, i, j)
    add rsp, 32              ; deallocate shadow shadowspace

    mov r8, r9               ; pass l to min call in r8
    pop rdx                  ; pop k(r8) to min call in rdx
    mov rcx, rax             ; pass min(g, i, j) to min call in rcx
    call min                 ; rax = min(min(g, i, j), k, l)
    ret 0
```

[t2.asm] - gcd



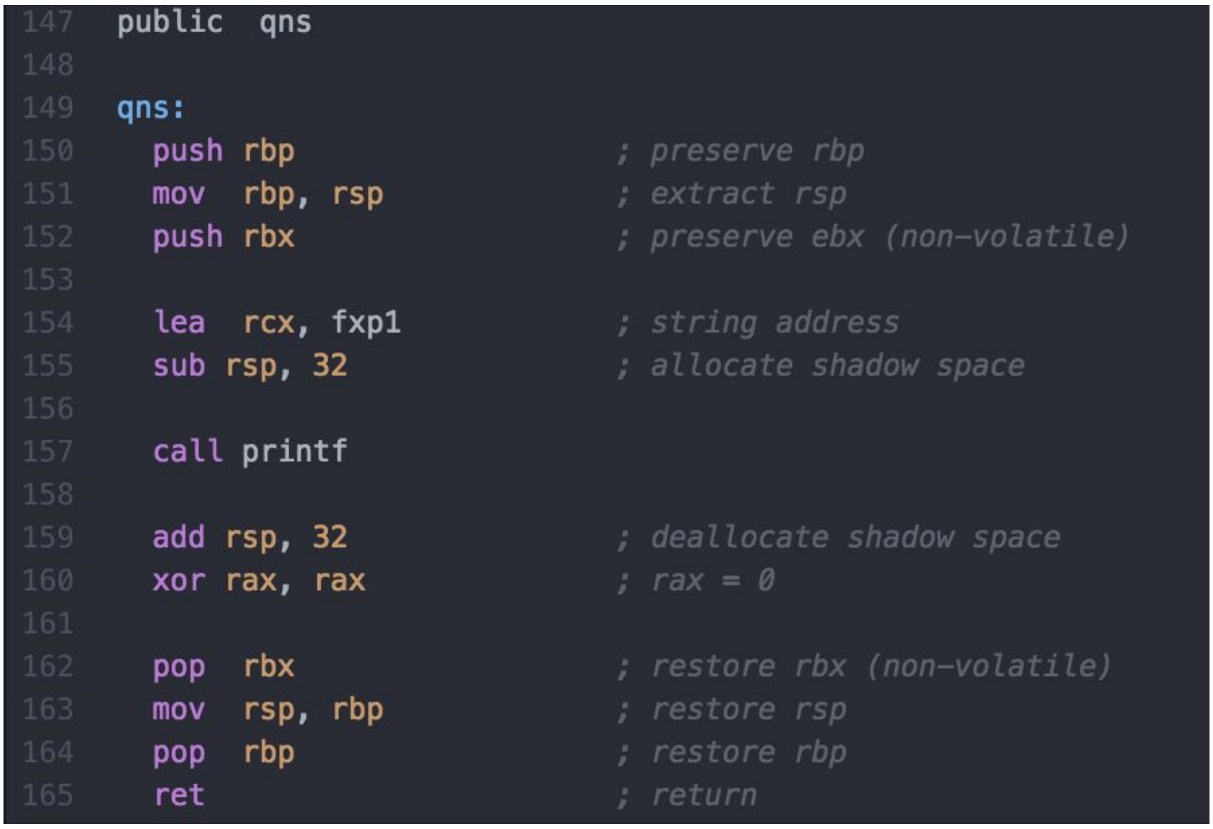
```asm
public  gcd                 ; export function name

gcd:
    cmp rdx, 0              ; if(b==0)
    je gcd_retA             ; return a

    mov rax, rcx            ; rax = a (dividend)
    mov rcx, rdx            ; rcx = b (divisor)
    cqo                     ; sign extend rax into rdx
    idiv rcx                ; rdx = a % b
    sub rsp, 32             ; allocate shadow space

    call gcd                ; rax = gcd(b, (a%b)) -> [b -> rcx, (a%b) -> rdx]

    add rsp, 32             ; deallocate shadow space
    jmp gcd_done            ;

gcd_retA:
    mov rax, rcx            ; rax = a

gcd_done:
    ret 0                   ; return

```

[t2.asm] - q



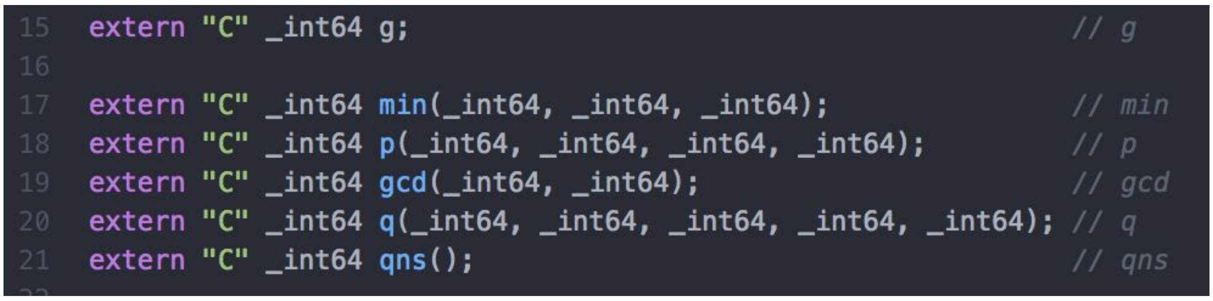
```
public   q                  ; export function name

q:
    push rbp
    mov  rbp, rsp           ; extract rsp
    push rbx                ; push non-volatile register


    lea rax,[rcx+rdx]       ; rax = a+b
    add rax,r8              ; rax += c
    add rax,r9              ; rax += d
    add rax,[rbp+48]        ; rax += e
    push rax                ; preserve sum (rax)

    push rax                ; push sum to printf stack
    push [rbp+48]           ; push e to printf stack
    push r9                 ; push d to printf stack
    mov  r9,  r8            ; r9 -> c
    mov  r8,  rdx           ; r8 -> b
    mov  rdx, rcx           ; rdx -> a
    lea  rcx, fxp2          ; rcx -> ascii format string address
    sub  rsp, 32            ; allocate shadowspace for printf

    call printf

    add  rsp, 32            ; de-allocate shadowspace
    add  rsp, 24            ; pop 3 (8 x 3) params off stack
    pop  rax                ; pop preserved sum off stack
    pop  rbx                ; restore rbx

    mov  rsp, rbp           ; restore rsp
    pop  rbp                ; restore rbp
    ret 0
```

[t2.asm] - qns



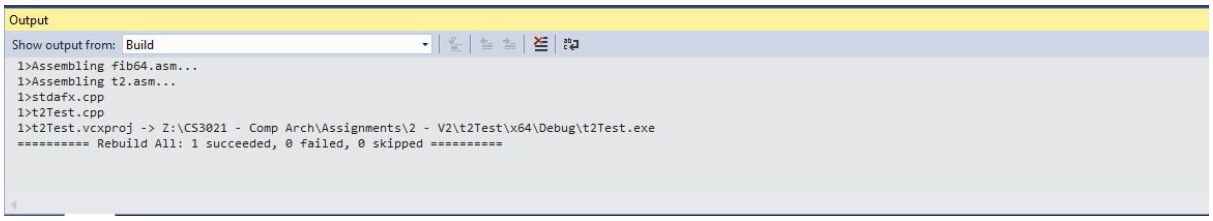
```
public  qns

qns:
  push rbp                        ; preserve rbp
  mov  rbp, rsp                   ; extract rsp
  push rbx                        ; preserve ebx (non-volatile)

  lea  rcx, fxp1                  ; string address
  sub rsp, 32                     ; allocate shadow space

  call printf

  add rsp, 32                     ; deallocate shadow space
  xor rax, rax                    ; rax = 0

  pop  rbx                        ; restore rbx (non-volatile)
  mov  rsp, rbp                   ; restore rsp
  pop  rbp                        ; restore rbp
  ret                             ; return
```

[t2.h]



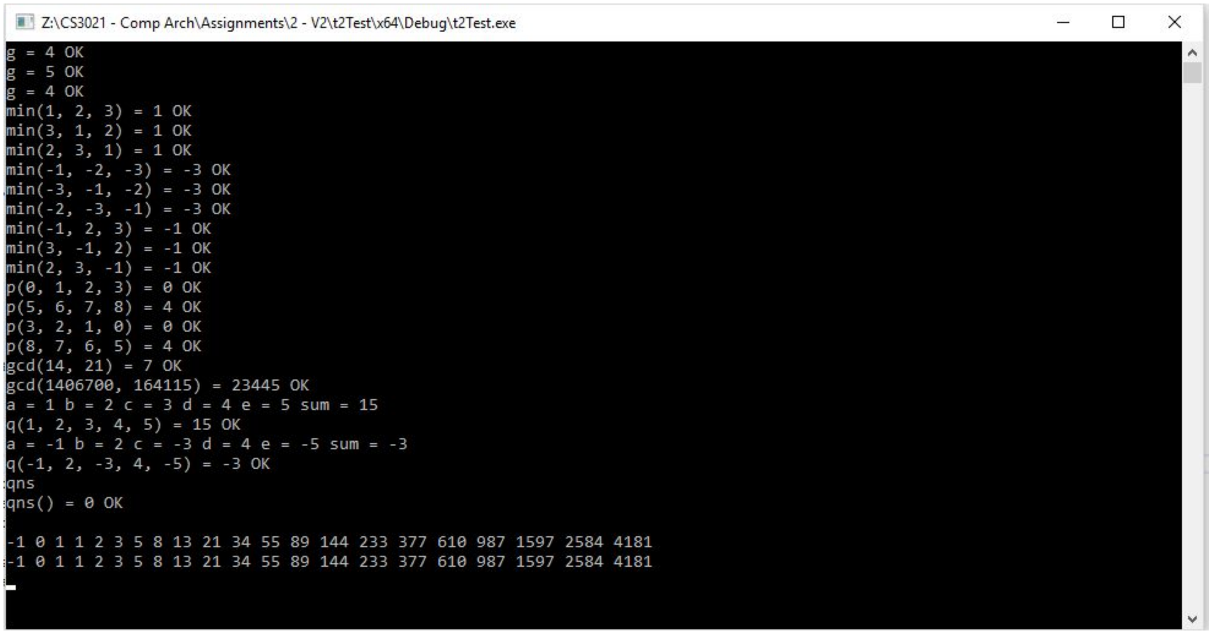
```
extern "C" _int64 g;                                               // g

extern "C" _int64 min(_int64, _int64, _int64);                     // min
extern "C" _int64 p(_int64, _int64, _int64, _int64);               // p
extern "C" _int64 gcd(_int64, _int64);                             // gcd
extern "C" _int64 q(_int64, _int64, _int64, _int64, _int64); // q
extern "C" _int64 qns();                                           // qns
```

3) Build Success



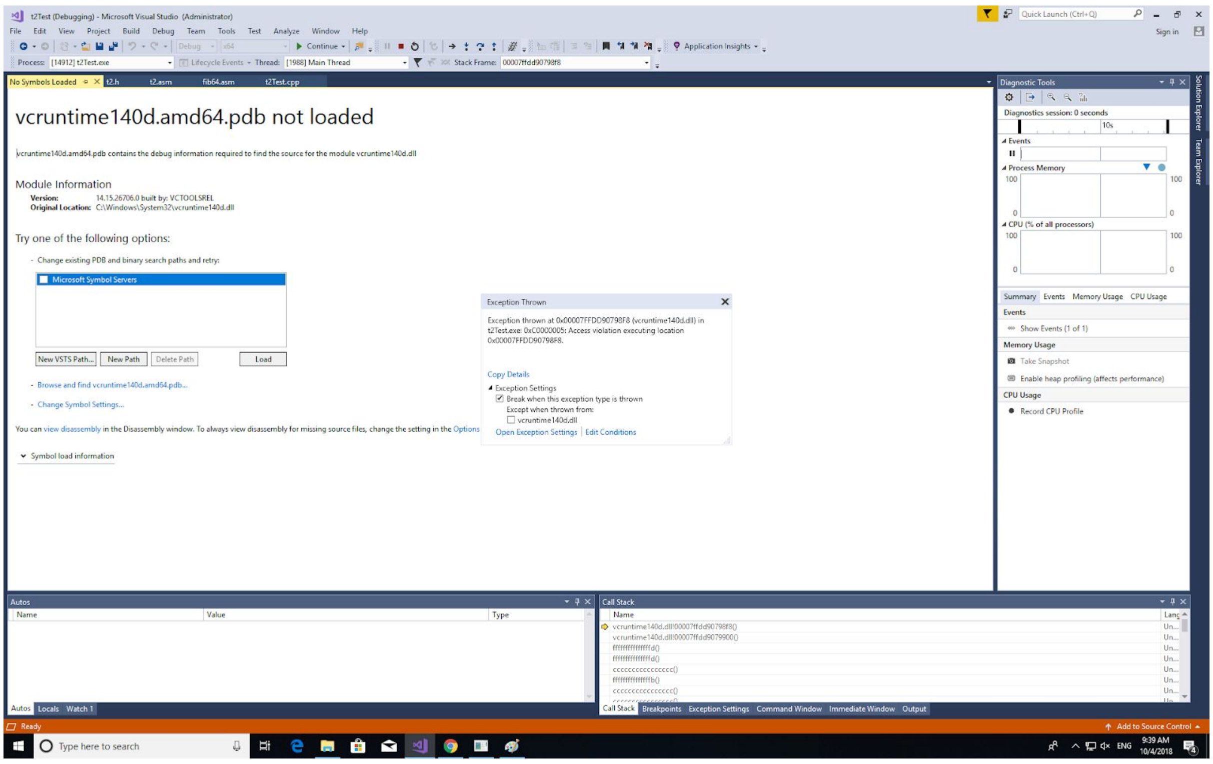
```
1>Assembling fib64.asm...
1>Assembling t2.asm...
1>stdafx.cpp
1>t2Test.cpp
1>t2Test.vcxproj -> Z:\CS3021 - Comp Arch\Assignments\2 - V2\t2Test\x64\Debug\t2Test.exe
========== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==========
```

3) Console Window (qns with Shadow Space)


Z:\CS3021 - Comp Arch\Assignments\2 - V2\t2Test\x64\Debug\t2Test.exe

```
g = 4 OK
g = 5 OK
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
a = 1 b = 2 c = 3 d = 4 e = 5 sum = 15
q(1, 2, 3, 4, 5) = 15 OK
a = -1 b = 2 c = -3 d = 4 e = -5 sum = -3
q(-1, 2, -3, 4, -5) = -3 OK
qns
qns() = 0 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
```

3) Qns without Shadow Space


t2Test (Debugging) - Microsoft Visual Studio (Administrator)

Process: [14912] t2Test.exe Thread: [1988] Main Thread Stack Frame: 00007ffdd90798f8

vcruntime140d.amd64.pdb not loaded

vcruntime140d.amd64.pdb contains the debug information required to find the source for the module vcruntime140d.dll

Module Information
Version: 14.15.26706.0 built by: VCTOOLSREL
Original Location: C:\Windows\System32\vcruntime140d.dll

Try one of the following options:
- Change existing PDB and binary search paths and retry:
- Browse and find vcruntime140d.amd64.pdb...
- Change Symbol Settings...

You can view disassembly in the Disassembly window. To always view disassembly for missing source files, change the setting in the Options

Exception Thrown

Exception thrown at 0x00007FFDD90798F8 (vcruntime140d.dll) in t2Test.exe: 0xC0000005: Access violation executing location 0x00007FFDD90798F8.

Exception Settings
[✓] Break when this exception type is thrown
Except when thrown from:
[ ] vcruntime140d.dll

Call Stack

```
vcruntime140d.dll!00007ffdd90798f8()
vcruntime140d.dll!00007ffdd9079900()
ffffffffffffffd()
ffffffffffffffd()
cccccccccccccccc()
ffffffffffffffb()
cccccccccccccccc()
```

When qns is executed without shadow space being allocated it causes an exception to be thrown as a result of an access violation when executing the instruction at location 0x00007FFDD….