

Take Home Test 2

MdShahid Bin Emdad

April 25th, 2022

CSC 34200

TABLE OF CONTENTS

Objective:.....	2
Description of Specifications, and Functionality:.....	2
PART I: MARS Simulator:.....	3
PART II: Visual Studio 2019 (Intel x86):.....	19
PART III: x86_64 ISA, Linux 64-bit:.....	23
Explanation:	26
Time-Comparison:	27
Conclusion:	28

I will neither give nor receive unauthorized assistance on this test.
 I will use only one computing device to perform this TEST, I will not use cell while performing this test

Shahid

start time and date: 04/23/22 10am

End time and date: 04/23/22 9:40pm

Objective:

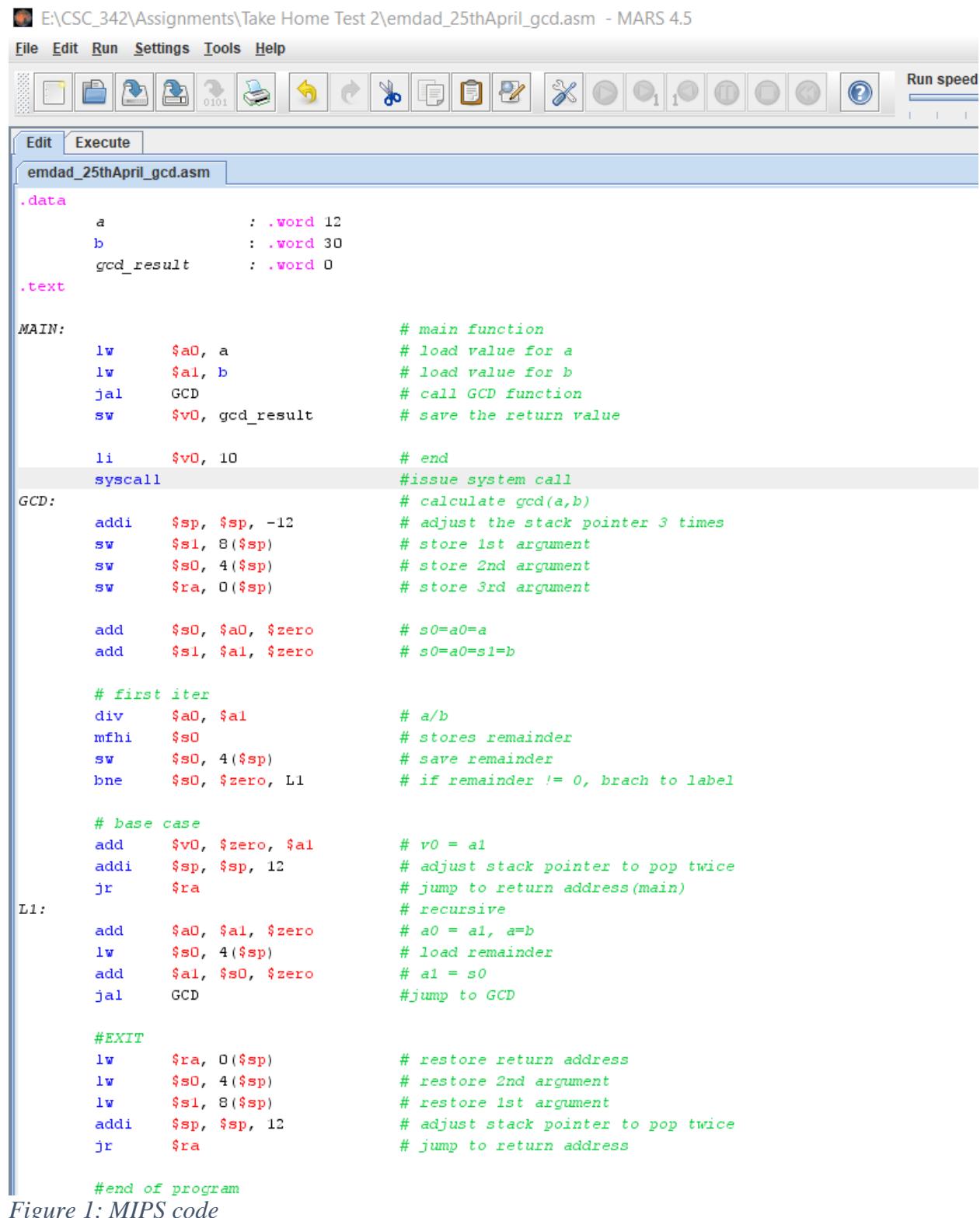
The objective of this test is to explore, understand and ability to demonstrate recursive calls and stack frames in the three sets of architecture. MIPS instructions set architecture, Intel x86 ISA using Windows MS 32-bit compiler and debugger and a Intel X86_64 bit ISA processor running Linux, 64 bit gcc and gdb. We used recursive version of EUCLIDEAN algorithm for two integers $a>0$, $b>0$.

Description of Specifications, and Functionality:

The digital system I used in this assignment is MARS to execute the assembly language and get the output, Microsoft Visual Studio 2019 to debug c programming languages program and get registers, disassemble and memory output. After that, I used Ubuntu Linux to run all the c files in that operating system and get output by using gdb. I used breakpoints while doing debugging.

PART I: MARS Simulator:

In figure 1, we can see the MIPS code on the MIPS simulator for recursive calls.



The screenshot shows the MARS 4.5 simulator interface with the assembly code for gcd.asm. The code implements a recursive GCD algorithm. It starts by loading values for *a* and *b*, calling the GCD function, and saving the result. It then issues a system call to end the program. The GCD function calculates the gcd(a,b) using the stack pointer (\$sp) to store arguments and temporary values. It performs iterative division until the remainder is zero, then handles the base case where *a* equals *b*. Finally, it restores the stack and returns to the main function.

```

E:\CSC_342\Assignments\Take Home Test 2\emdad_25thApril_gcd.asm - MARS 4.5

File Edit Run Settings Tools Help
Run speed

Edit Execute
emdad_25thApril_gcd.asm

.data
    a : .word 12
    b : .word 30
    gcd_result : .word 0

.text
MAIN:
    lw      $a0, a          # main function
    lw      $a1, b          # load value for a
    jal     GCD             # call GCD function
    sw      $v0, gcd_result # save the return value

    li      $v0, 10          # end
    syscall                   #issue system call

GCD:
    addi   $sp, $sp, -12      # calculate gcd(a,b)
    sw      $s1, 8($sp)       # adjust the stack pointer 3 times
    sw      $s0, 4($sp)       # store 1st argument
    sw      $ra, 0($sp)       # store 2nd argument
    sw      $ra, 0($sp)       # store 3rd argument

    add    $s0, $a0, $zero    # s0=a0=a
    add    $s1, $a1, $zero    # s0=a0=s1=b

    # first iter
    div    $a0, $a1           # a/b
    mfhi   $s0               # stores remainder
    sw      $s0, 4($sp)       # save remainder
    bne   $s0, $zero, L1      # if remainder != 0, brach to label

    # base case
    add    $v0, $zero, $a1     # v0 = a1
    addi  $sp, $sp, 12         # adjust stack pointer to pop twice
    jr    $ra                  # jump to return address(main)

L1:
    add    $a0, $a1, $zero     # a0 = a1, a=b
    lw     $s0, 4($sp)        # load remainder
    add    $a1, $s0, $zero     # a1 = s0
    jal     GCD              #jump to GCD

    #EXIT
    lw      $ra, 0($sp)        # restore return address
    lw      $s0, 4($sp)        # restore 2nd argument
    lw      $s1, 8($sp)        # restore 1st argument
    addi  $sp, $sp, 12         # adjust stack pointer to pop twice
    jr    $ra                  # jump to return address

#end of program

```

Figure 1: MIPS code

In figure 2, we can see that the operation completed successfully.

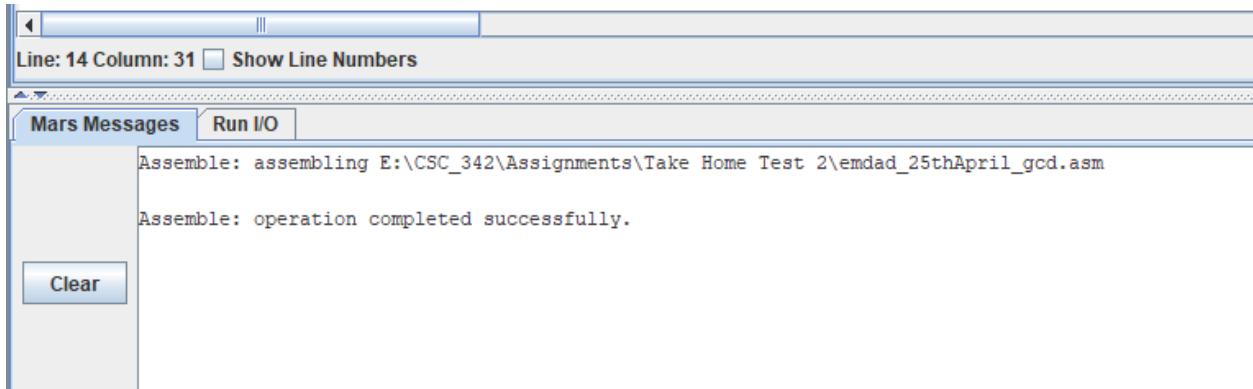


Figure 2: operation completion report

In figure 3, we can see the text segment, data segment and registers, labels for the MIPS code after execution.

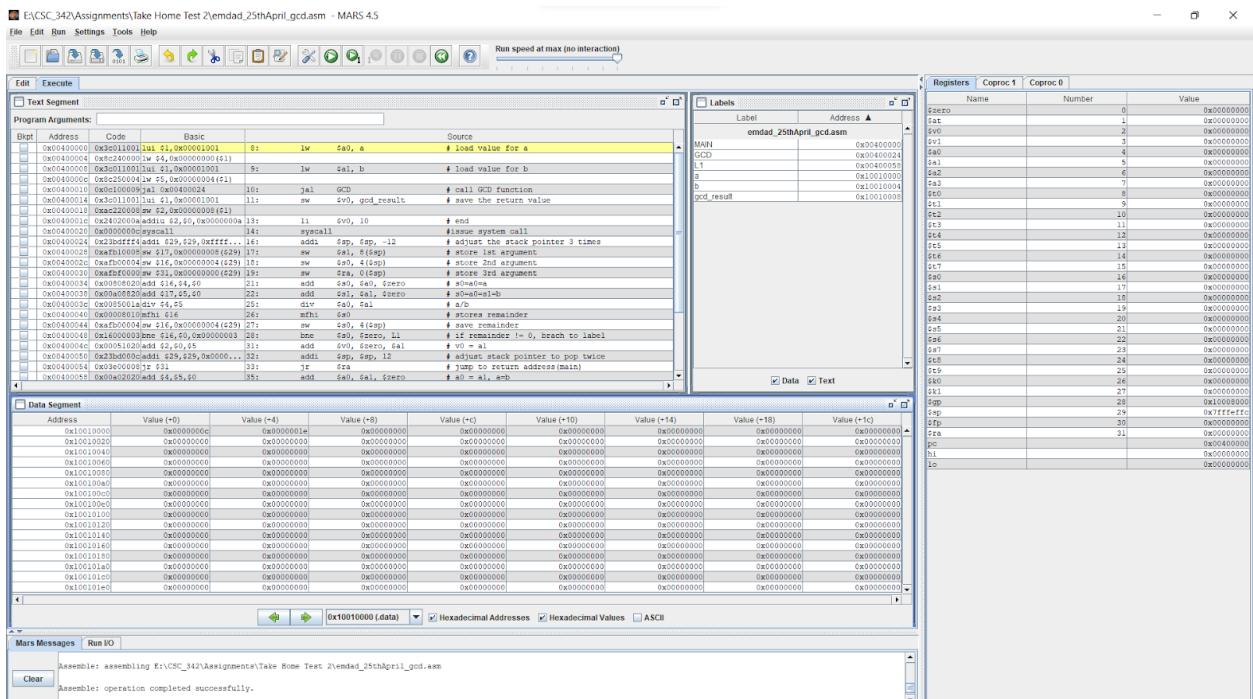


Figure 3: text segment, data segment and registers window

In figure 4, we can see the value for a stored in \$a0.

The screenshot shows the MARS assembly editor interface with the following details:

- Assembly View (Left):** Displays the assembly code for `emddad_25thApril_gcd.asm`. The code implements a GCD function using a recursive algorithm. It uses registers `s0`, `s1`, and `s2` for arguments and local variables, and `r0` for the result. The stack pointer `sp` is used to manage the call stack.
- Registers View (Top Right):** Shows the state of various registers:
 - Coproc 1:** `sp` = 0x00000000, `bp` = 0x00000000, `lr` = 0x00000000, `pc` = 0x00000000, `r0` = 0x00000000, `r1` = 0x00000000, `r2` = 0x00000000, `r3` = 0x00000000, `r4` = 0x00000000, `r5` = 0x00000000, `r6` = 0x00000000, `r7` = 0x00000000, `r8` = 0x00000000, `r9` = 0x00000000, `r10` = 0x00000000, `r11` = 0x00000000, `r12` = 0x00000000, `r13` = 0x00000000, `r14` = 0x00000000, `r15` = 0x00000000, `r16` = 0x00000000, `r17` = 0x00000000, `r18` = 0x00000000, `r19` = 0x00000000, `r20` = 0x00000000, `r21` = 0x00000000, `r22` = 0x00000000, `r23` = 0x00000000, `r24` = 0x00000000, `r25` = 0x00000000, `r26` = 0x00000000, `r27` = 0x00000000, `r28` = 0x00000000, `r29` = 0x00000000, `r30` = 0x00000000, `r31` = 0x00000000.
 - Coproc 0:** `sp` = 0x00000000, `bp` = 0x00000000, `lr` = 0x00000000, `pc` = 0x00000000, `r0` = 0x00000000, `r1` = 0x00000000, `r2` = 0x00000000, `r3` = 0x00000000, `r4` = 0x00000000, `r5` = 0x00000000, `r6` = 0x00000000, `r7` = 0x00000000, `r8` = 0x00000000, `r9` = 0x00000000, `r10` = 0x00000000, `r11` = 0x00000000, `r12` = 0x00000000, `r13` = 0x00000000, `r14` = 0x00000000, `r15` = 0x00000000, `r16` = 0x00000000, `r17` = 0x00000000, `r18` = 0x00000000, `r19` = 0x00000000, `r20` = 0x00000000, `r21` = 0x00000000, `r22` = 0x00000000, `r23` = 0x00000000, `r24` = 0x00000000, `r25` = 0x00000000, `r26` = 0x00000000, `r27` = 0x00000000, `r28` = 0x00000000, `r29` = 0x00000000, `r30` = 0x00000000, `r31` = 0x00000000.
- Labels View (Top Right):** Lists the labels defined in the program: `MAIN`, `GCD`, `L1`, `a`, `b`, `gcd_result`.
- Data Segment View (Bottom):** Shows the memory dump for the data segment, listing addresses from 0x00100000 to 0x00100100 with their corresponding values.

Figure 4: value of a stored in \$a0

In figure 5, we can see the value for b stored in \$a0.

Figure 5: value for b stored in \$a0.

In figure 6, we can see that it is back to main function after assigning the values into the variables.

<img alt="Screenshot of MARS 4.5 assembly editor showing the state of the program after variable assignments. The registers window shows \$zero=0, \$v0=10, \$v1=2, \$a0=1, \$a1=2, \$t0=4, \$t1=5, \$t2=6, \$t3=7, \$t4=8, \$t5=9, \$t6=10, \$t7=11, \$t8=12, \$t9=13, \$t10=14, \$t11=15, \$t12=16, \$t13=17, \$t14=18, \$t15=19, \$t16=20, \$t17=21, \$t18=22, \$t19=23, \$t20=24, \$t21=25, \$t22=26, \$t23=27, \$t24=28, \$t25=29, \$t26=30, \$t27=31, \$t28=32, \$t29=33, \$t30=34, \$t31=35, \$t32=36, \$t33=37, \$t34=38, \$t35=39, \$t36=40, \$t37=41, \$t38=42, \$t39=43, \$t40=44, \$t41=45, \$t42=46, \$t43=47, \$t44=48, \$t45=49, \$t46=50, \$t47=51, \$t48=52, \$t49=53, \$t50=54, \$t51=55, \$t52=56, \$t53=57, \$t54=58, \$t55=59, \$t56=60, \$t57=61, \$t58=62, \$t59=63, \$t60=64, \$t61=65, \$t62=66, \$t63=67, \$t64=68, \$t65=69, \$t66=70, \$t67=71, \$t68=72, \$t69=73, \$t70=74, \$t71=75, \$t72=76, \$t73=77, \$t74=78, \$t75=79, \$t76=80, \$t77=81, \$t78=82, \$t79=83, \$t80=84, \$t81=85, \$t82=86, \$t83=87, \$t84=88, \$t85=89, \$t86=90, \$t87=91, \$t88=92, \$t89=93, \$t90=94, \$t91=95, \$t92=96, \$t93=97, \$t94=98, \$t95=99, \$t96=100, \$t97=101, \$t98=102, \$t99=103, \$t100=104, \$t101=105, \$t102=106, \$t103=107, \$t104=108, \$t105=109, \$t106=110, \$t107=111, \$t108=112, \$t109=113, \$t110=114, \$t111=115, \$t112=116, \$t113=117, \$t114=118, \$t115=119, \$t116=120, \$t117=121, \$t118=122, \$t119=123, \$t120=124, \$t121=125, \$t122=126, \$t123=127, \$t124=128, \$t125=129, \$t126=130, \$t127=131, \$t128=132, \$t129=133, \$t130=134, \$t131=135, \$t132=136, \$t133=137, \$t134=138, \$t135=139, \$t136=140, \$t137=141, \$t138=142, \$t139=143, \$t140=144, \$t141=145, \$t142=146, \$t143=147, \$t144=148, \$t145=149, \$t146=150, \$t147=151, \$t148=152, \$t149=153, \$t150=154, \$t151=155, \$t152=156, \$t153=157, \$t154=158, \$t155=159, \$t156=160, \$t157=161, \$t158=162, \$t159=163, \$t160=164, \$t161=165, \$t162=166, \$t163=167, \$t164=168, \$t165=169, \$t166=170, \$t167=171, \$t168=172, \$t169=173, \$t170=174, \$t171=175, \$t172=176, \$t173=177, \$t174=178, \$t175=179, \$t176=180, \$t177=181, \$t178=182, \$t179=183, \$t180=184, \$t181=185, \$t182=186, \$t183=187, \$t184=188, \$t185=189, \$t186=190, \$t187=191, \$t188=192, \$t189=193, \$t190=194, \$t191=195, \$t192=196, \$t193=197, \$t194=198, \$t195=199, \$t196=200, \$t197=201, \$t198=202, \$t199=203, \$t200=204, \$t201=205, \$t202=206, \$t203=207, \$t204=208, \$t205=209, \$t206=210, \$t207=211, \$t208=212, \$t209=213, \$t210=214, \$t211=215, \$t212=216, \$t213=217, \$t214=218, \$t215=219, \$t216=220, \$t217=221, \$t218=222, \$t219=223, \$t220=224, \$t221=225, \$t222=226, \$t223=227, \$t224=228, \$t225=229, \$t226=230, \$t227=231, \$t228=232, \$t229=233, \$t230=234, \$t231=235, \$t232=236, \$t233=237, \$t234=238, \$t235=239, \$t236=240, \$t237=241, \$t238=242, \$t239=243, \$t240=244, \$t241=245, \$t242=246, \$t243=247, \$t244=248, \$t245=249, \$t246=250, \$t247=251, \$t248=252, \$t249=253, \$t250=254, \$t251=255, \$t252=256, \$t253=257, \$t254=258, \$t255=259, \$t256=260, \$t257=261, \$t258=262, \$t259=263, \$t260=264, \$t261=265, \$t262=266, \$t263=267, \$t264=268, \$t265=269, \$t266=270, \$t267=271, \$t268=272, \$t269=273, \$t270=274, \$t271=275, \$t272=276, \$t273=277, \$t274=278, \$t275=279, \$t276=280, \$t277=281, \$t278=282, \$t279=283, \$t280=284, \$t281=285, \$t282=286, \$t283=287, \$t284=288, \$t285=289, \$t286=290, \$t287=291, \$t288=292, \$t289=293, \$t290=294, \$t291=295, \$t292=296, \$t293=297, \$t294=298, \$t295=299, \$t296=300, \$t297=301, \$t298=302, \$t299=303, \$t300=304, \$t301=305, \$t302=306, \$t303=307, \$t304=308, \$t305=309, \$t306=310, \$t307=311, \$t308=312, \$t309=313, \$t310=314, \$t311=315, \$t312=316, \$t313=317, \$t314=318, \$t315=319, \$t316=320, \$t317=321, \$t318=322, \$t319=323, \$t320=324, \$t321=325, \$t322=326, \$t323=327, \$t324=328, \$t325=329, \$t326=330, \$t327=331, \$t328=332, \$t329=333, \$t330=334, \$t331=335, \$t332=336, \$t333=337, \$t334=338, \$t335=339, \$t336=340, \$t337=341, \$t338=342, \$t339=343, \$t340=344, \$t341=345, \$t342=346, \$t343=347, \$t344=348, \$t345=349, \$t346=350, \$t347=351, \$t348=352, \$t349=353, \$t350=354, \$t351=355, \$t352=356, \$t353=357, \$t354=358, \$t355=359, \$t356=360, \$t357=361, \$t358=362, \$t359=363, \$t360=364, \$t361=365, \$t362=366, \$t363=367, \$t364=368, \$t365=369, \$t366=370, \$t367=371, \$t368=372, \$t369=373, \$t370=374, \$t371=375, \$t372=376, \$t373=377, \$t374=378, \$t375=379, \$t376=380, \$t377=381, \$t378=382, \$t379=383, \$t380=384, \$t381=385, \$t382=386, \$t383=387, \$t384=388, \$t385=389, \$t386=390, \$t387=391, \$t388=392, \$t389=393, \$t390=394, \$t391=395, \$t392=396, \$t393=397, \$t394=398, \$t395=399, \$t396=400, \$t397=401, \$t398=402, \$t399=403, \$t400=404, \$t401=405, \$t402=406, \$t403=407, \$t404=408, \$t405=409, \$t406=410, \$t407=411, \$t408=412, \$t409=413, \$t410=414, \$t411=415, \$t412=416, \$t413=417, \$t414=418, \$t415=419, \$t416=420, \$t417=421, \$t418=422, \$t419=423, \$t420=424, \$t421=425, \$t422=426, \$t423=427, \$t424=428, \$t425=429, \$t426=430, \$t427=431, \$t428=432, \$t429=433, \$t430=434, \$t431=435, \$t432=436, \$t433=437, \$t434=438, \$t435=439, \$t436=440, \$t437=441, \$t438=442, \$t439=443, \$t440=444, \$t441=445, \$t442=446, \$t443=447, \$t444=448, \$t445=449, \$t446=450, \$t447=451, \$t448=452, \$t449=453, \$t450=454, \$t451=455, \$t452=456, \$t453=457, \$t454=458, \$t455=459, \$t456=460, \$t457=461, \$t458=462, \$t459=463, \$t460=464, \$t461=465, \$t462=466, \$t463=467, \$t464=468, \$t465=469, \$t466=470, \$t467=471, \$t468=472, \$t469=473, \$t470=474, \$t471=475, \$t472=476, \$t473=477, \$t474=478, \$t475=479, \$t476=480, \$t477=481, \$t478=482, \$t479=483, \$t480=484, \$t481=485, \$t482=486, \$t483=487, \$t484=488, \$t485=489, \$t486=490, \$t487=491, \$t488=492, \$t489=493, \$t490=494, \$t491=495, \$t492=496, \$t493=497, \$t494=498, \$t495=499, \$t496=500, \$t497=501, \$t498=502, \$t499=503, \$t500=504, \$t501=505, \$t502=506, \$t503=507, \$t504=508, \$t505=509, \$t506=510, \$t507=511, \$t508=512, \$t509=513, \$t510=514, \$t511=515, \$t512=516, \$t513=517, \$t514=518, \$t515=519, \$t516=520, \$t517=521, \$t518=522, \$t519=523, \$t520=524, \$t521=525, \$t522=526, \$t523=527, \$t524=528, \$t525=529, \$t526=530, \$t527=531, \$t528=532, \$t529=533, \$t530=534, \$t531=535, \$t532=536, \$t533=537, \$t534=538, \$t535=539, \$t536=540, \$t537=541, \$t538=542, \$t539=543, \$t540=544, \$t541=545, \$t542=546, \$t543=547, \$t544=548, \$t545=549, \$t546=550, \$t547=551, \$t548=552, \$t549=553, \$t550=554, \$t551=555, \$t552=556, \$t553=557, \$t554=558, \$t555=559, \$t556=560, \$t557=561, \$t558=562, \$t559=563, \$t560=564, \$t561=565, \$t562=566, \$t563=567, \$t564=568, \$t565=569, \$t566=570, \$t567=571, \$t568=572, \$t569=573, \$t570=574, \$t571=575, \$t572=576, \$t573=577, \$t574=578, \$t575=579, \$t576=580, \$t577=581, \$t578=582, \$t579=583, \$t580=584, \$t581=585, \$t582=586, \$t583=587, \$t584=588, \$t585=589, \$t586=590, \$t587=591, \$t588=592, \$t589=593, \$t590=594, \$t591=595, \$t592=596, \$t593=597, \$t594=598, \$t595=599, \$t596=600, \$t597=601, \$t598=602, \$t599=603, \$t600=604, \$t601=605, \$t602=606, \$t603=607, \$t604=608, \$t605=609, \$t606=610, \$t607=611, \$t608=612, \$t609=613, \$t610=614, \$t611=615, \$t612=616, \$t613=617, \$t614=618, \$t615=619, \$t616=620, \$t617=621, \$t618=622, \$t619=623, \$t620=624, \$t621=625, \$t622=626, \$t623=627, \$t624=628, \$t625=629, \$t626=630, \$t627=631, \$t628=632, \$t629=633, \$t630=634, \$t631=635, \$t632=636, \$t633=637, \$t634=638, \$t635=639, \$t636=640, \$t637=641, \$t638=642, \$t639=643, \$t640=644, \$t641=645, \$t642=646, \$t643=647, \$t644=648, \$t645=649, \$t646=650, \$t647=651, \$t648=652, \$t649=653, \$t650=654, \$t651=655, \$t652=656, \$t653=657, \$t654=658, \$t655=659, \$t656=660, \$t657=661, \$t658=662, \$t659=663, \$t660=664, \$t661=665, \$t662=666, \$t663=667, \$t664=668, \$t665=669, \$t666=670, \$t667=671, \$t668=672, \$t669=673, \$t670=674, \$t671=675, \$t672=676, \$t673=677, \$t674=678, \$t675=679, \$t676=680, \$t677=681, \$t678=682, \$t679=683, \$t680=684, \$t681=685, \$t682=686, \$t683=687, \$t684=688, \$t685=689, \$t686=690, \$t687=691, \$t688=692, \$t689=693, \$t690=694, \$t691=695, \$t692=696, \$t693=697, \$t694=698, \$t695=699, \$t696=700, \$t697=701, \$t698=702, \$t699=703, \$t700=704, \$t691=705, \$t692=706, \$t693=707, \$t694=708, \$t695=709, \$t696=710, \$t697=711, \$t698=712, \$t699=713, \$t700=714, \$t691=715, \$t692=716, \$t693=717, \$t694=718, \$t695=719, \$t696=720, \$t697=721, \$t698=722, \$t699=723, \$t700=724, \$t691=725, \$t692=726, \$t693=727, \$t694=728, \$t695=729, \$t696=730, \$t697=731, \$t698=732, \$t699=733, \$t700=734, \$t691=735, \$t692=736, \$t693=737, \$t694=738, \$t695=739, \$t696=740, \$t697=741, \$t698=742, \$t699=743, \$t700=744, \$t691=745, \$t692=746, \$t693=747, \$t694=748, \$t695=749, \$t696=750, \$t697=751, \$t698=752, \$t699=753, \$t700=754, \$t691=755, \$t692=756, \$t693=757, \$t694=758, \$t695=759, \$t696=760, \$t697=761, \$t698=762, \$t699=763, \$t700=764, \$t691=765, \$t692=766, \$t693=767, \$t694=768, \$t695=769, \$t696=770, \$t697=771, \$t698=772, \$t699=773, \$t700=774, \$t691=775, \$t692=776, \$t693=777, \$t694=778, \$t695=779, \$t696=780, \$t697=781, \$t698=782, \$t699=783, \$t700=784, \$t691=785, \$t692=786, \$t693=787, \$t694=788, \$t695=789, \$t696=790, \$t697=791, \$t698=792, \$t699=793, \$t700=794, \$t691=795, \$t692=796, \$t693=797, \$t694=798, \$t695=799, \$t696=800, \$t697=801, \$t698=802, \$t699=803, \$t700=804, \$t691=805, \$t692=806, \$t693=807, \$t694=808, \$t695=809, \$t696=810, \$t697=811, \$t698=812, \$t699=813, \$t700=814, \$t691=815, \$t692=816, \$t693=817, \$t694=818, \$t695=819, \$t696=820, \$t697=821, \$t698=822, \$t699=823, \$t700=824, \$t691=825, \$t692=826, \$t693=827, \$t694=828, \$t695=829, \$t696=830, \$t697=831, \$t698=832, \$t699=833, \$t700=834, \$t691=835, \$t692=836, \$t693=837, \$t694=838, \$t695=839, \$t696=840, \$t697=841, \$t698=842, \$t699=843, \$t700=844, \$t691=845, \$t692=846, \$t693=847, \$t694=848, \$t695=849, \$t696=850, \$t697=851, \$t698=852, \$t699=853, \$t700=854, \$t691=855, \$t692=856, \$t693=857, \$t694=858, \$t695=859, \$t696=860, \$t697=861, \$t698=862, \$t699=863, \$t700=864, \$t691=865, \$t692=866, \$t693=867, \$t694=868, \$t695=869, \$t696=870, \$t697=871, \$t698=872, \$t699=873, \$t700=874, \$t691=875, \$t692=876, \$t693=877, \$t694=878, \$t695=879, \$t696=880, \$t697=881, \$t698=882, \$t699=883, \$t700=884, \$t691=885, \$t692=886, \$t693=887, \$t694=888, \$t695=889, \$t696=890, \$t697=891, \$t698=892, \$t699=893, \$t700=894, \$t691=895, \$t692=896, \$t693=897, \$t694=898, \$t695=899, \$t696=900, \$t697=901, \$t698=902, \$t699=903, \$t700=904, \$t691=905, \$t692=906, \$t693=907, \$t694=908, \$t695=909, \$t696=910, \$t697=911, \$t698=912, \$t699=913, \$t700=914, \$t691=915, \$t692=916, \$t693=917, \$t694=918, \$t695=919, \$t696=920, \$t697=921, \$t698=922, \$t699=923, \$t700=924, \$t691=925, \$t692=926, \$t693=927, \$t694=928, \$t695=929, \$t696=930, \$t697=931, \$t698=932, \$t699=933, \$t700=934, \$t691=935, \$t692=936, \$t693=937, \$t694=938, \$t695=939, \$t696=940, \$t697=941, \$t698=942, \$t699=943, \$t700=944, \$t691=945, \$t692=946, \$t693=947, \$t694=948, \$t695=949, \$t696=950, \$t697=951, \$t698=952, \$t699=953, \$t700=954, \$t691=955, \$t692=956, \$t693=957, \$t694=958, \$t695=959, \$t696=960, \$t697=961, \$t698=962, \$t699=963, \$t700=964, \$t691=965, \$t692=966, \$t693=967, \$t694=968, \$t695=969, \$t696=970, \$t697=971, \$t698=972, \$t699=973, \$t700=974, \$t691=975, \$t692=976, \$t693=977, \$t694=978, \$t695=979, \$t696=980, \$t697=981, \$t698=982, \$t699=983, \$t700=984, \$t691=985, \$t692=986, \$t693=987, \$t694=988, \$t695=989, \$t696=990, \$t697=991, \$t698=992, \$t699=993, \$t700=994, \$t691=995, \$t692=996, \$t693=997, \$t694=998, \$t695=999, \$t696=1000, \$t697=1001, \$t698=1002, \$t699=1003, \$t700=1004, \$t691=1005, \$t692=1006, \$t693=1007, \$t694=1008, \$t695=1009, \$t696=1010, \$t697=1011, \$t698=1012, \$t699=1013, \$t700=1014, \$t691=1015, \$t692=1016, \$t693=1017, \$t694=1018, \$t695=1019, \$t696=1020, \$t697=1021, \$t698=1022, \$t699=1023, \$t700=1024, \$t691=1025, \$t692=1026, \$t693=1027, \$t694=1028, \$t695=1029, \$t696=1030, \$t697=1031, \$t698=1032, \$t699=1033, \$t700=1034, \$t691=1035, \$t692=1036, \$t693=1037, \$t694=1038, \$t695=1039, \$t696=1040, \$t697=1041, \$t698=1042, \$t699=1043, \$t700=1044, \$t691=1045, \$t692=1046, \$t693=1047, \$t694=1048, \$t695=1049, \$t696=1050, \$t697=1051, \$t698=1052, \$t699=1053, \$t700=1054, \$t691=1055, \$t692=1056, \$t693=1057, \$t694=1058, \$t695=1059, \$t696=1060, \$t697=1061, \$t698=1062, \$t699=1063, \$t700=1064, \$t691=1065, \$t692=1066, \$t693=1067, \$t694=1068, \$t695=1069, \$t696=1070, \$t697=1071, \$t698=1072, \$t699=1073, \$t700=1074, \$t691=1075, \$t692=1076, \$t693=1077, \$t694=1078, \$t695=1079, \$t696=1080, \$t697=1081, \$t698=1082, \$t699=1083, \$t700=1084, \$t691=1085, \$t692=1086, \$t693=1087, \$t694=1088, \$t695=1089, \$t696=1090, \$t697=1091, \$t698=1092, \$t699=1093, \$t700=1094, \$t691=1095, \$t692=1096, \$t693=1097, \$t694=1098, \$t695=1099, \$t696=1100, \$t697=1101, \$t698=1102, \$t699=1103, \$t700=1104, \$t691=1105, \$t692=1106, \$t693=1107, \$t694=1108, \$t695=1109, \$t696=1110, \$t697=1111, \$t698=1112, \$t699=1113, \$t700=1114, \$t691=1115, \$t692=1116, \$t693=1117, \$t694=1118, \$t695=1119, \$t696=1120, \$t697=1121, \$t698=1122, \$t699=1123, \$t700=1124, \$t691=1125, \$t692=1126, \$t693=1127, \$t694=1128, \$t695=1129, \$t696=1130, \$t697=1131, \$t698

In figure 8, we can see that, the stack pointer moved and the address for \$ra is 0x00400014.

The screenshot shows the MARS 4.5 assembly editor interface. The top menu bar includes File, Edit, Run, Settings, Tools, Help, and a Run speed at max (no interaction) button. The main window has three main panes: Registers, Code Segment, and Data Segment.

Registers:

Name	Number	Value
szero	0	0x00000000
sat	1	0x10010000
sr	2	0x00000000
sp	3	0x00000000
sr0	4	0x00000000
sr1	5	0x00000000
sr2	6	0x00000000
sr3	7	0x00000000
sr4	8	0x00000000
sr5	9	0x00000000
sr6	10	0x00000000
sr7	11	0x00000000
sr8	12	0x00000000
sr9	13	0x00000000
sr10	14	0x00000000
sr11	15	0x00000000
sr12	16	0x00000000
sr13	17	0x00000000
sr14	18	0x00000000
sr15	19	0x00000000
sr16	20	0x00000000
sr17	21	0x00000000
sr18	22	0x00000000
sr19	23	0x00000000
sr20	24	0x00000000
sr21	25	0x00000000
sr22	26	0x00000000
sr23	27	0x00000000
sr24	28	0x00000000
sr25	29	0x00000000
sr26	30	0x00000000
sr27	31	0x00000000
pc	32	0x00000000
sr28	33	0x00000000
sr29	34	0x00000000
sr30	35	0x00000000
sr31	36	0x00000000
sr32	37	0x00000000
sr33	38	0x00000000
sr34	39	0x00000000
sr35	40	0x00000000
sr36	41	0x00000000
sr37	42	0x00000000
sr38	43	0x00000000
sr39	44	0x00000000
sr40	45	0x00000000
sr41	46	0x00000000
sr42	47	0x00000000
sr43	48	0x00000000
sr44	49	0x00000000
sr45	50	0x00000000
sr46	51	0x00000000
sr47	52	0x00000000
sr48	53	0x00000000
sr49	54	0x00000000
sr50	55	0x00000000
sr51	56	0x00000000
sr52	57	0x00000000
sr53	58	0x00000000
sr54	59	0x00000000
sr55	60	0x00000000
sr56	61	0x00000000
sr57	62	0x00000000
sr58	63	0x00000000
sr59	64	0x00000000
sr60	65	0x00000000
sr61	66	0x00000000
sr62	67	0x00000000
sr63	68	0x00000000
sr64	69	0x00000000
sr65	70	0x00000000
sr66	71	0x00000000
sr67	72	0x00000000
sr68	73	0x00000000
sr69	74	0x00000000
sr70	75	0x00000000
sr71	76	0x00000000
sr72	77	0x00000000
sr73	78	0x00000000
sr74	79	0x00000000
sr75	80	0x00000000
sr76	81	0x00000000
sr77	82	0x00000000
sr78	83	0x00000000
sr79	84	0x00000000
sr80	85	0x00000000
sr81	86	0x00000000
sr82	87	0x00000000
sr83	88	0x00000000
sr84	89	0x00000000
sr85	90	0x00000000
sr86	91	0x00000000
sr87	92	0x00000000
sr88	93	0x00000000
sr89	94	0x00000000
sr90	95	0x00000000
sr91	96	0x00000000
sr92	97	0x00000000
sr93	98	0x00000000
sr94	99	0x00000000
sr95	100	0x00000000
sr96	101	0x00000000
sr97	102	0x00000000
sr98	103	0x00000000
sr99	104	0x00000000
sr100	105	0x00000000
sr101	106	0x00000000
sr102	107	0x00000000
sr103	108	0x00000000
sr104	109	0x00000000
sr105	110	0x00000000
sr106	111	0x00000000
sr107	112	0x00000000
sr108	113	0x00000000
sr109	114	0x00000000
sr110	115	0x00000000
sr111	116	0x00000000
sr112	117	0x00000000
sr113	118	0x00000000
sr114	119	0x00000000
sr115	120	0x00000000
sr116	121	0x00000000
sr117	122	0x00000000
sr118	123	0x00000000
sr119	124	0x00000000
sr120	125	0x00000000
sr121	126	0x00000000
sr122	127	0x00000000
sr123	128	0x00000000
sr124	129	0x00000000
sr125	130	0x00000000
sr126	131	0x00000000
sr127	132	0x00000000
sr128	133	0x00000000
sr129	134	0x00000000
sr130	135	0x00000000
sr131	136	0x00000000
sr132	137	0x00000000
sr133	138	0x00000000
sr134	139	0x00000000
sr135	140	0x00000000
sr136	141	0x00000000
sr137	142	0x00000000
sr138	143	0x00000000
sr139	144	0x00000000
sr140	145	0x00000000
sr141	146	0x00000000
sr142	147	0x00000000
sr143	148	0x00000000
sr144	149	0x00000000
sr145	150	0x00000000
sr146	151	0x00000000
sr147	152	0x00000000
sr148	153	0x00000000
sr149	154	0x00000000
sr150	155	0x00000000
sr151	156	0x00000000
sr152	157	0x00000000
sr153	158	0x00000000
sr154	159	0x00000000
sr155	160	0x00000000
sr156	161	0x00000000
sr157	162	0x00000000
sr158	163	0x00000000
sr159	164	0x00000000
sr160	165	0x00000000
sr161	166	0x00000000
sr162	167	0x00000000
sr163	168	0x00000000
sr164	169	0x00000000
sr165	170	0x00000000
sr166	171	0x00000000
sr167	172	0x00000000
sr168	173	0x00000000
sr169	174	0x00000000
sr170	175	0x00000000
sr171	176	0x00000000
sr172	177	0x00000000
sr173	178	0x00000000
sr174	179	0x00000000
sr175	180	0x00000000
sr176	181	0x00000000
sr177	182	0x00000000
sr178	183	0x00000000
sr179	184	0x00000000
sr180	185	0x00000000
sr181	186	0x00000000
sr182	187	0x00000000
sr183	188	0x00000000
sr184	189	0x00000000
sr185	190	0x00000000
sr186	191	0x00000000
sr187	192	0x00000000
sr188	193	0x00000000
sr189	194	0x00000000
sr190	195	0x00000000
sr191	196	0x00000000
sr192	197	0x00000000
sr193	198	0x00000000
sr194	199	0x00000000
sr195	200	0x00000000
sr196	201	0x00000000
sr197	202	0x00000000
sr198	203	0x00000000
sr199	204	0x00000000
sr200	205	0x00000000
sr201	206	0x00000000
sr202	207	0x00000000
sr203	208	0x00000000
sr204	209	0x00000000
sr205	210	0x00000000
sr206	211	0x00000000
sr207	212	0x00000000
sr208	213	0x00000000
sr209	214	0x00000000
sr210	215	0x00000000
sr211	216	0x00000000
sr212	217	0x00000000
sr213	218	0x00000000
sr214	219	0x00000000
sr215	220	0x00000000
sr216	221	0x00000000
sr217	222	0x00000000
sr218	223	0x00000000
sr219	224	0x00000000
sr220	225	0x00000000
sr221	226	0x00000000
sr222	227	0x00000000
sr223	228	0x00000000
sr224	229	0x00000000
sr225	230	0x00000000
sr226	231	0x00000000
sr227	232	0x00000000
sr228	233	0x00000000
sr229	234	0x00000000
sr230	235	0x00000000
sr231	236	0x00000000
sr232	237	0x00000000
sr233	238	0x00000000
sr234	239	0x00000000
sr235	240	0x00000000
sr236	241	0x00000000
sr237	242	0x00000000
sr238	243	0x00000000
sr239	244	0x00000000
sr240	245	0x00000000
sr241	246	0x00000000
sr242	247	0x00000000
sr243	248	0x00000000
sr244	249	0x00000000
sr245	250	0x00000000
sr246	251	0x00000000
sr247	252	0x00000000
sr248	253	0x00000000
sr249	254	0x00000000
sr250	255	0x00000000
sr251	256	0x00000000
sr252	257	0x00000000
sr253	258	0x00000000
sr254	259	0x00000000
sr255	260	0x00000000
sr256	261	0x00000000
sr257	262	0x00000000
sr258	263	0x00000000
sr259	264	0x00000000
sr260	265	0x00000000
sr261	266	0x00000000
sr262	267	0x00000000
sr263	268	0x00000000
sr264	269	0x00000000
sr265	270	0x00000000
sr266	271	0x00000000
sr267	272	0x00000000
sr268	273	0x00000000
sr269	274	0x00000000
sr270	275	0x00000000
sr271	276	0x00000000
sr272	277	0x00000000
sr273	278	0x00000000
sr274	279	0x00000000
sr275	280	0x00000000
sr276	281	0x00000000
sr277	282	0x00000000
sr278	283	0x00000000
sr279	284	0x00000000
sr280	285	0x00000000
sr281	286	0x00000000
sr282	287	0x00000000
sr283	288	0x00000000
sr284	289	0x00000000
sr285	290	0x00000000
sr286	291	0x00000000
sr287	292	0x00000000
sr288	293	0x00000000
sr289	294	0x00000000
sr290	295	0x00000000
sr291	296	0x00000000
sr292	297	0x00000000
sr293	298	0x00000000
sr294	299	0x00000000
sr295	300	0x00000000
sr296	301	0x00000000
sr297	302	0x00000000
sr298	303	0x00000000
sr299	304	0x00000000
sr300	305	0x00000000
sr301	306	0x00000000
sr302	307	0x00000000
sr303	308	0x00000000
sr304	309	0x00000000
sr305	310	0x00000000
sr306	311	0x00000000
sr307	312	0x00000000
sr308	313	0x00000000
sr309	314	0x00000000
sr310	315	0x00000000
sr311	316	0x00000000
sr312	317	0x00000000
sr313	318	0x00000000
sr314	319	0x00000000
sr315	320	0x00000000
sr316	321	0x00000000
sr317	322	0x00000000
sr318	323	0x00000000
sr319	324	0x00000000
sr320	325	0x00000000
sr321	326	0x00000000
sr322	327	0x00000000
sr323	328	0x00000000
sr324	329	0x00000000
sr325	330	0x00000000
sr326	331	0x00000000
sr327	332	0x00000000
sr328	333	0x00000000
sr329	334	0x00000000
sr330	335	0x00000000
sr331	336	0x00000000
sr332	337	0x00000000
sr333	338	0x00000000
sr334	339	0x00000000
sr335	340	0x00000000
sr336	341	0x00000000
sr337	342	0x00000000
sr338	343	0x000

Figure 8: stack pointer adjusted

In figure 9, we can see that \$s1 is storing the 1st argument. The address for \$sp is 0x7ffffe0.

Figure 9: stored 1st argument

In figure 10, we can see that \$s0 is storing the 2nd argument and \$sp address is still same. The offset is +18 and the address value is 0.

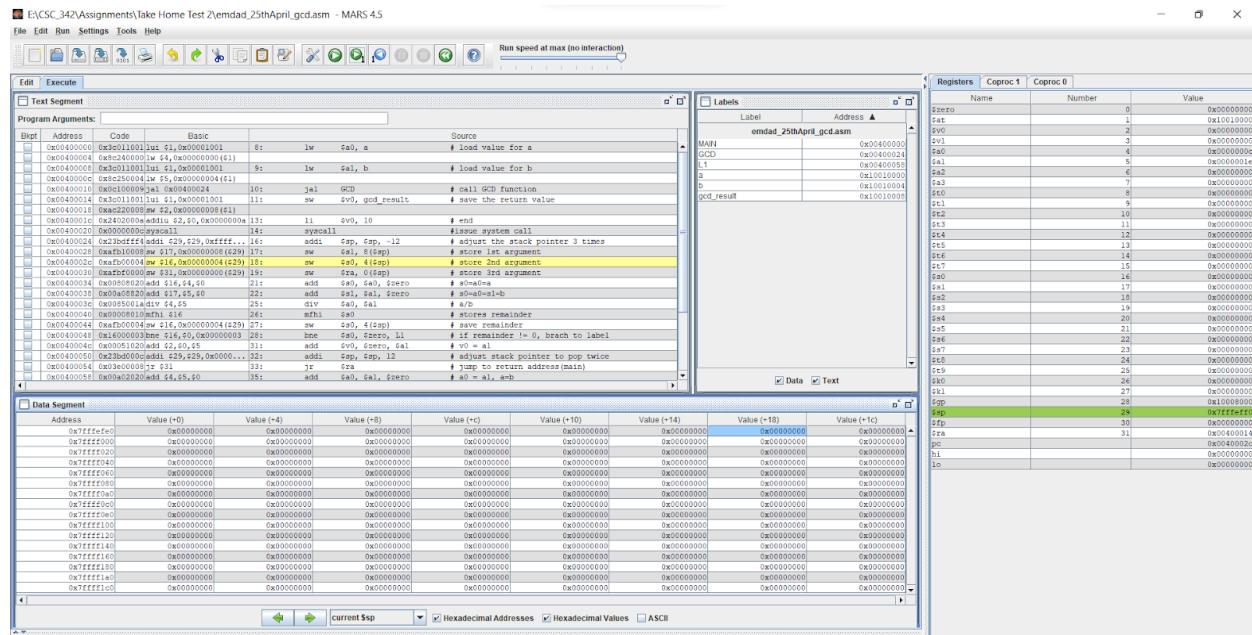


Figure 10: 2nd argument storing

In figure 11, we can see that \$ra is storing the 3rd argument and \$sp address is still same. The offset is +14 and the address, 0x7fffffe0 value is 0.

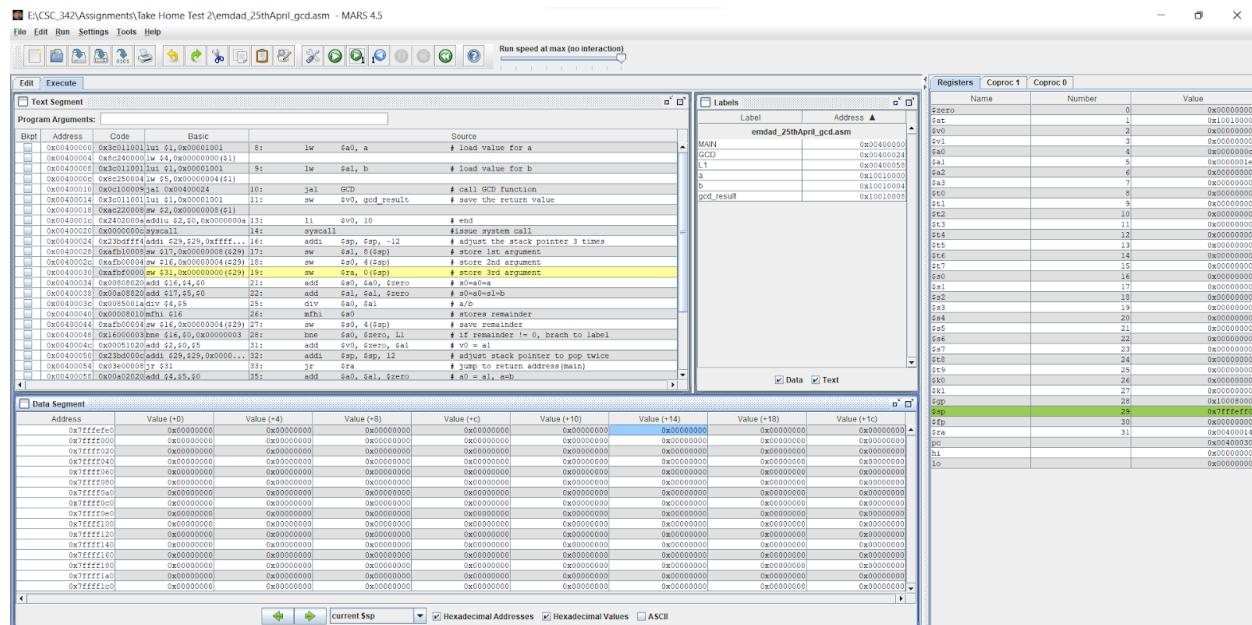


Figure 11: storing the 3rd argument

In figure 12, we can see that the replacing value happening on line 21, means $s0=a0=a$. The $\$sp$ address is still but the offset is +10 and value for the address, $0x7ffffe0$ is $0x0040014$.

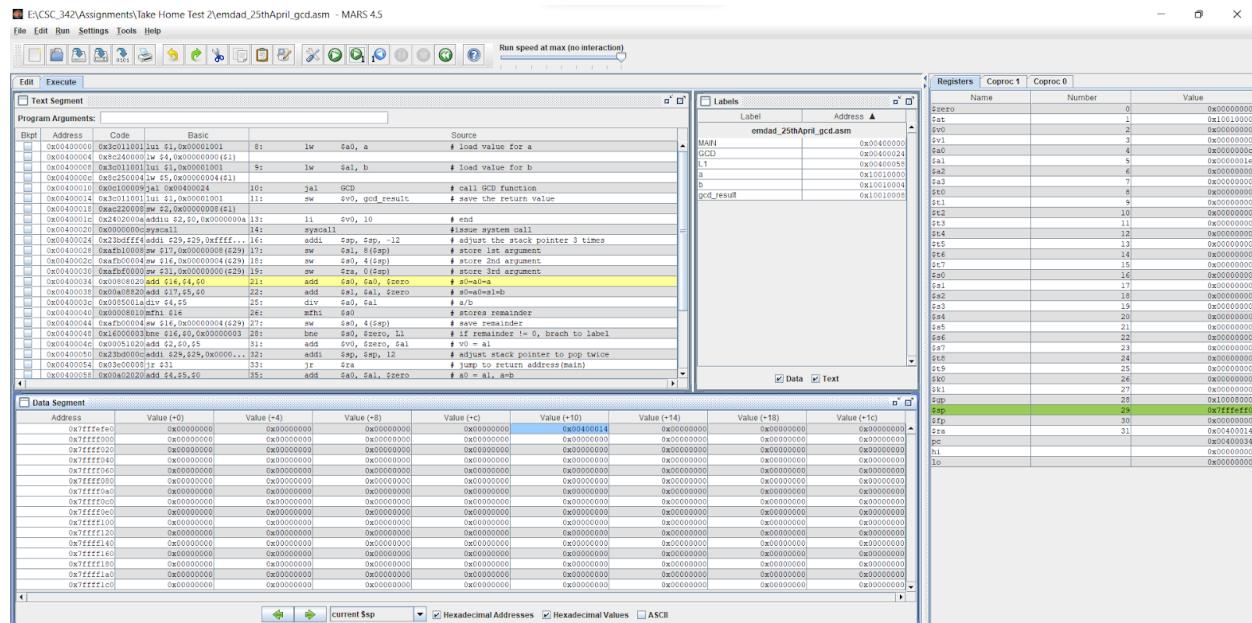


Figure 12: add \$s0, \$a0

In figure 13, we can see that the program moved to line 22 and now it is replacing value \$s1 with \$a1 which means s0=a0=s1=b. The register address for \$s0 is 0x000000c.

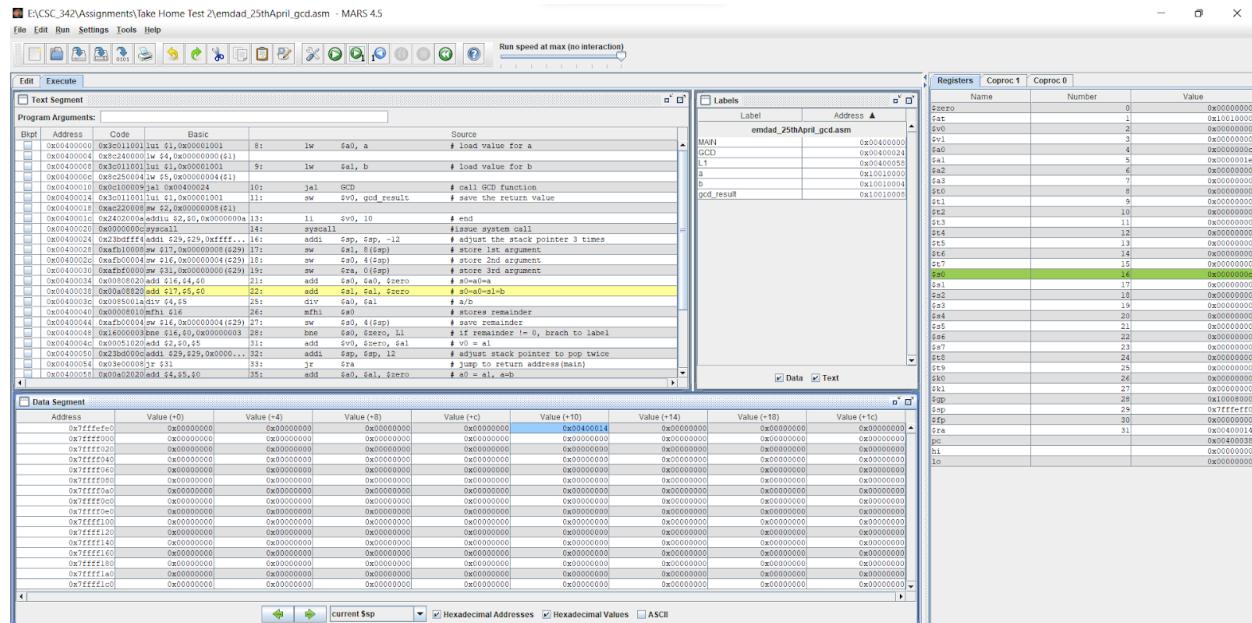


Figure 13: add \$s1, \$a1

In figure 14, we can see that it is doing division. We are diving a by b. and the stored address is \$s1, 0x0000001e and the memory offset is +10 and value is 0x00400014.

The screenshot shows the MARS 4.5 assembly editor interface. The assembly code for `emdad_25thApril_gcd.asm` is displayed in the left pane, with various assembly instructions and their comments. The right pane shows the memory dump, registers, and stack. The bottom navigation bar includes tabs for Current Ssp, Hexadecimal Addresses, Hexadecimal Values, and ASCII.

Figure 14: division

In figure 15, we can see that “*mfhi*” is storing the remainder. The register name is *lo*, and value is 0x00000000.

Figure 15: storing remainder

In figure 16, we can see that \$s0 is saving the remainder and the register, \$s0 and value is 0x0000000c.

The screenshot shows the MARS 4.5 assembly editor with the following details:

- Registers:** \$zero = 0, \$t0 = 1, \$s0 = 2, \$v1 = 3, \$s1 = 4, \$s2 = 5, \$s3 = 6, \$s4 = 7, \$s5 = 8, \$s6 = 9, \$s7 = 10, \$s8 = 11, \$s9 = 12, \$s10 = 13, \$s11 = 14, \$s12 = 15, \$s13 = 16, \$s14 = 17, \$s15 = 18, \$s16 = 19, \$s17 = 20, \$s18 = 21, \$s19 = 22, \$s20 = 23, \$s21 = 24, \$s22 = 25, \$s23 = 26, \$s24 = 27, \$s25 = 28, \$s26 = 29, \$s27 = 30, \$t1 = 31.
- Labels:** MAIN, GCD, L1, b, d, gcd_result.
- Registers window:** Shows the value of \$s0 as 0x0000000c.
- Data Segment:** Shows memory starting at address 0x7fffffe0 with all values set to 0x00000000.

Figure 16: save remainder into \$s0

In figure 17, we can see that the bne computation is happening and it is checking if the remainder != 0, if so, it will branch to label. The memory address offset is +14 and value stores is 0x000000c.

The screenshot shows the MARS 4.5 assembly editor with the following details:

- Registers:** \$zero = 0, \$t0 = 1, \$s0 = 2, \$v1 = 3, \$s1 = 4, \$s2 = 5, \$s3 = 6, \$s4 = 7, \$s5 = 8, \$s6 = 9, \$s7 = 10, \$s8 = 11, \$s9 = 12, \$s10 = 13, \$s11 = 14, \$s12 = 15, \$s13 = 16, \$s14 = 17, \$s15 = 18, \$s16 = 19, \$s17 = 20, \$s18 = 21, \$s19 = 22, \$s20 = 23, \$s21 = 24, \$s22 = 25, \$s23 = 26, \$s24 = 27, \$s25 = 28, \$s26 = 29, \$s27 = 30, \$t1 = 31.
- Labels:** MAIN, GCD, L1, b, d, gcd_result.
- Registers window:** Shows the value of \$s0 as 0x0000000c.
- Data Segment:** Shows memory starting at address 0x7fffffe0 with all values set to 0x00000000.

Figure 17: BNE computation

In figure 18, we can see that code went to line 35 to do $a_0 = a_1$ and $a=b$. The memory address value stays same.

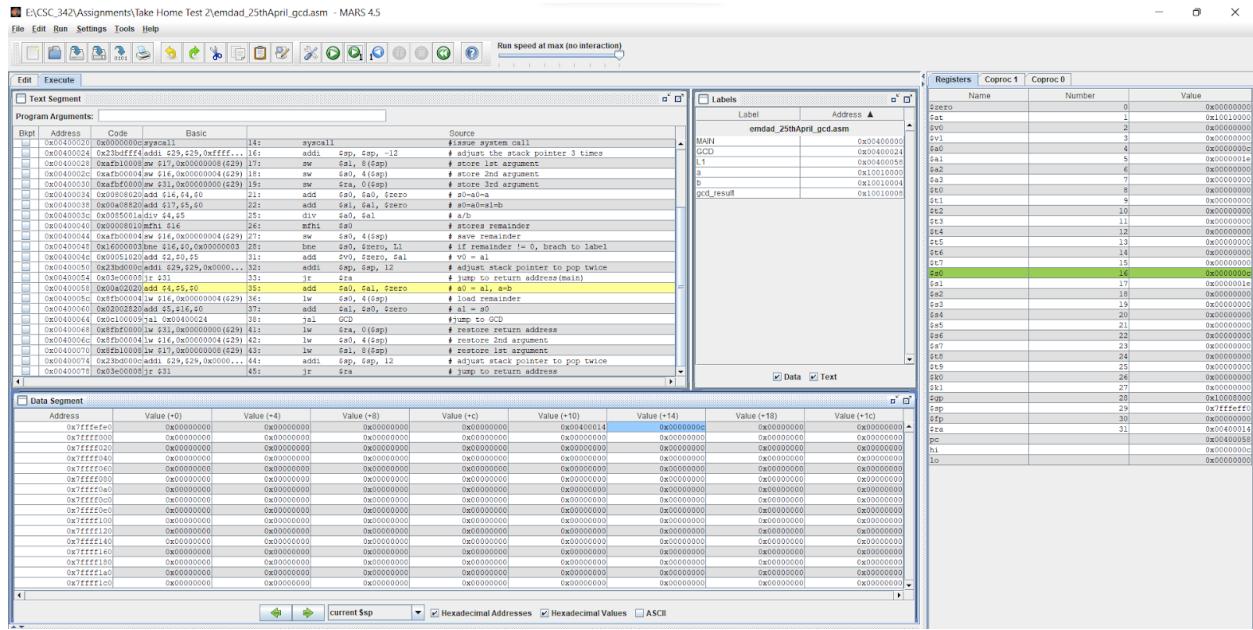


Figure 18: add \$a0, \$a1

In figure 19, we can see that line 36 is loading the remainder and the register moved to \$a0 which value is 0x00001e. The offset value is 0 for +14.

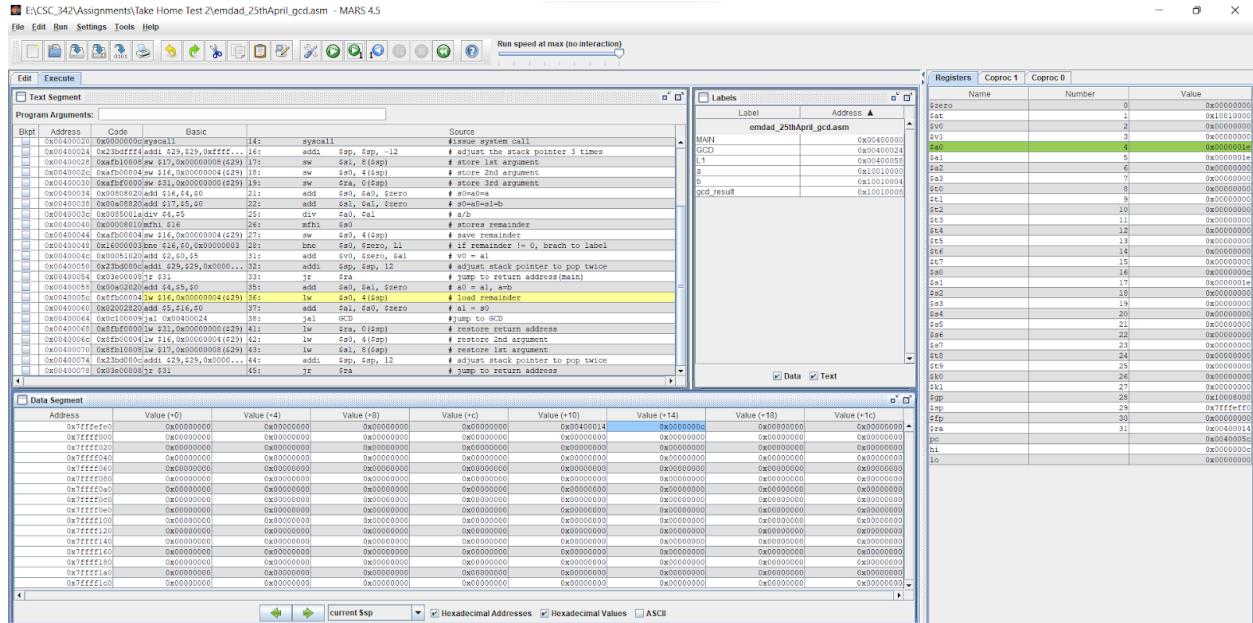


Figure 19: loading remainder

In figure 20, we can see that the values of \$a0 is switching to \$a1 and the value for register \$s0 is 0x0000000c.

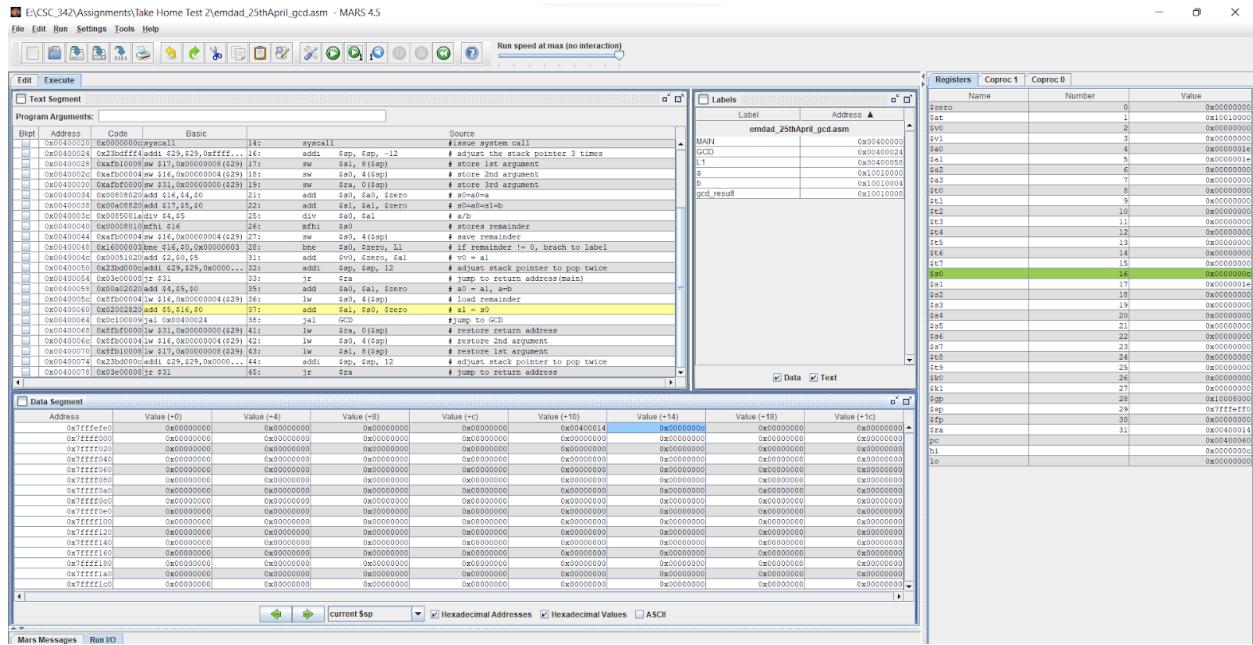


Figure 20: add \$a1, \$a0

In figure 21, we can see that the program jumps to GCD again.

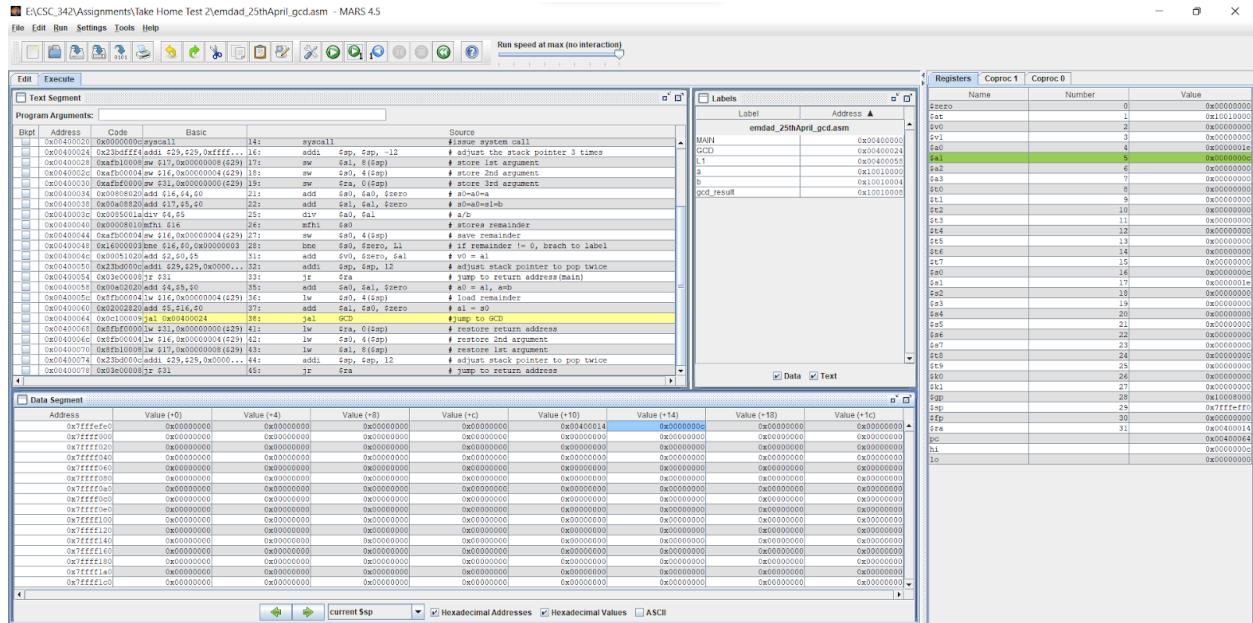


Figure 21: jump to gcd

In figure 22, we can see that we are adjusting the stack pointer now.

```

E:\CSC_342\Assignments\Take Home Test\emdad_25thApril_gcd.asm - MARS 4.5
File Edit Run Settings Tools Help
Run speed at max (no interaction)
Edit Execute
Text Segment
Program Arguments:
Bfrt Address Code Basic Source
0x0400024 0x230fffff addi $29,$29,0xfffffe... 14: syscall # issue system call
0x0400025 0x230fffff addi $29,$29,0xfffffe... 15: addi $sp,$sp,-12 # adjust the stack pointer 3 times
0x0400026 0x230fffff sw $17,$0($sp) # store 1st argument
0x0400027 0x230fffff sw $18,$1($sp) # store 2nd argument
0x0400028 0x230fffff sw $19,$2($sp) # store 3rd argument
0x0400029 0x230fffff sw $21,$3($sp) # store 4th argument
0x0400030 0x230fffff addi $16,$4,00 21: add $a1,$a1,$zero # $a1=a1
0x0400031 0x230fffff addi $17,$5,00 22: add $a2,$a2,$zero # $a2=a2
0x0400032 0x230fffff addi $18,$6,00 23: add $a3,$a3,$zero # $a3=a3
0x0400033 0x230fffff addi $19,$7,00 24: add $a4,$a4,$zero # $a4=a4
0x0400034 0x230fffff addi $16,$8,00 25: add $a5,$a5,$zero # $a5=a5
0x0400035 0x230fffff addi $17,$9,00 26: add $a6,$a6,$zero # $a6=a6
0x0400036 0x230fffff addi $18,$10,00 27: add $a7,$a7,$zero # $a7=a7
0x0400037 0x230fffff addi $19,$11,00 28: add $a8,$a8,$zero # $a8=a8
0x0400038 0x230fffff addi $20,$12,00 29: add $a9,$a9,$zero # $a9=a9
0x0400039 0x230fffff addi $21,$13,00 30: add $a10,$a10,$zero # $a10=a10
0x0400040 0x230fffff addi $22,$14,00 31: add $a11,$a11,$zero # $a11=a11
0x0400041 0x230fffff addi $23,$15,00 32: add $a12,$a12,$zero # $a12=a12
0x0400042 0x230fffff addi $24,$16,00 33: add $a13,$a13,$zero # $a13=a13
0x0400043 0x230fffff addi $25,$17,00 34: add $a14,$a14,$zero # $a14=a14
0x0400044 0x230fffff addi $26,$18,00 35: add $a15,$a15,$zero # $a15=a15
0x0400045 0x230fffff addi $27,$19,00 36: add $a16,$a16,$zero # $a16=a16
0x0400046 0x230fffff addi $28,$20,00 37: add $a17,$a17,$zero # $a17=a17
0x0400047 0x230fffff addi $29,$21,00 38: jal $C0 # jump to GCD
0x0400048 0x230fffff addi $29,$22,00 39: add $a18,$a18,$zero # $a18=a18
0x0400049 0x230fffff addi $29,$23,00 40: add $a19,$a19,$zero # $a19=a19
0x040004a 0x230fffff addi $29,$24,00 41: add $a20,$a20,$zero # $a20=a20
0x040004b 0x230fffff addi $29,$25,00 42: add $a21,$a21,$zero # $a21=a21
0x040004c 0x230fffff addi $29,$26,00 43: add $a22,$a22,$zero # $a22=a22
0x040004d 0x230fffff addi $29,$27,00 44: add $a23,$a23,$zero # $a23=a23
0x040004e 0x230fffff addi $29,$28,00 45: add $a24,$a24,$zero # $a24=a24
0x040004f 0x230fffff addi $29,$29,00 46: add $a25,$a25,$zero # $a25=a25
0x0400050 0x230fffff addi $29,$29,00 47: add $a26,$a26,$zero # $a26=a26
0x0400051 0x230fffff addi $29,$29,00 48: add $a27,$a27,$zero # $a27=a27
0x0400052 0x230fffff addi $29,$29,00 49: add $a28,$a28,$zero # $a28=a28
0x0400053 0x230fffff addi $29,$29,00 50: add $a29,$a29,$zero # $a29=a29
0x0400054 0x230fffff addi $29,$29,00 51: add $a30,$a30,$zero # $a30=a30
0x0400055 0x230fffff addi $29,$29,00 52: add $a31,$a31,$zero # $a31=a31
0x0400056 0x230fffff addi $29,$29,00 53: add $a32,$a32,$zero # $a32=a32
0x0400057 0x230fffff addi $29,$29,00 54: add $a33,$a33,$zero # $a33=a33
0x0400058 0x230fffff addi $29,$29,00 55: add $a34,$a34,$zero # $a34=a34
0x0400059 0x230fffff addi $29,$29,00 56: add $a35,$a35,$zero # $a35=a35
0x040005a 0x230fffff addi $29,$29,00 57: add $a36,$a36,$zero # $a36=a36
0x040005b 0x230fffff addi $29,$29,00 58: add $a37,$a37,$zero # $a37=a37
0x040005c 0x230fffff addi $29,$29,00 59: add $a38,$a38,$zero # $a38=a38
0x040005d 0x230fffff addi $29,$29,00 60: add $a39,$a39,$zero # $a39=a39
0x040005e 0x230fffff addi $29,$29,00 61: add $a40,$a40,$zero # $a40=a40
0x040005f 0x230fffff addi $29,$29,00 62: add $a41,$a41,$zero # $a41=a41
0x0400060 0x230fffff addi $29,$29,00 63: add $a42,$a42,$zero # $a42=a42
0x0400061 0x230fffff addi $29,$29,00 64: add $a43,$a43,$zero # $a43=a43
0x0400062 0x230fffff addi $29,$29,00 65: add $a44,$a44,$zero # $a44=a44
0x0400063 0x230fffff addi $29,$29,00 66: add $a45,$a45,$zero # $a45=a45
0x0400064 0x230fffff addi $29,$29,00 67: add $a46,$a46,$zero # $a46=a46
0x0400065 0x230fffff addi $29,$29,00 68: add $a47,$a47,$zero # $a47=a47
0x0400066 0x230fffff addi $29,$29,00 69: add $a48,$a48,$zero # $a48=a48
0x0400067 0x230fffff addi $29,$29,00 70: add $a49,$a49,$zero # $a49=a49
0x0400068 0x230fffff addi $29,$29,00 71: add $a50,$a50,$zero # $a50=a50
0x0400069 0x230fffff addi $29,$29,00 72: add $a51,$a51,$zero # $a51=a51
0x040006a 0x230fffff addi $29,$29,00 73: add $a52,$a52,$zero # $a52=a52
0x040006b 0x230fffff addi $29,$29,00 74: add $a53,$a53,$zero # $a53=a53
0x040006c 0x230fffff addi $29,$29,00 75: add $a54,$a54,$zero # $a54=a54
0x040006d 0x230fffff addi $29,$29,00 76: add $a55,$a55,$zero # $a55=a55
0x040006e 0x230fffff addi $29,$29,00 77: add $a56,$a56,$zero # $a56=a56
0x040006f 0x230fffff addi $29,$29,00 78: add $a57,$a57,$zero # $a57=a57
0x0400070 0x230fffff addi $29,$29,00 79: add $a58,$a58,$zero # $a58=a58
0x0400071 0x230fffff addi $29,$29,00 80: add $a59,$a59,$zero # $a59=a59
0x0400072 0x230fffff addi $29,$29,00 81: add $a60,$a60,$zero # $a60=a60
0x0400073 0x230fffff addi $29,$29,00 82: add $a61,$a61,$zero # $a61=a61
0x0400074 0x230fffff addi $29,$29,00 83: add $a62,$a62,$zero # $a62=a62
0x0400075 0x230fffff addi $29,$29,00 84: add $a63,$a63,$zero # $a63=a63
0x0400076 0x230fffff addi $29,$29,00 85: add $a64,$a64,$zero # $a64=a64
0x0400077 0x230fffff addi $29,$29,00 86: add $a65,$a65,$zero # $a65=a65
0x0400078 0x230fffff addi $29,$29,00 87: add $a66,$a66,$zero # $a66=a66
0x0400079 0x230fffff addi $29,$29,00 88: add $a67,$a67,$zero # $a67=a67
0x040007a 0x230fffff addi $29,$29,00 89: add $a68,$a68,$zero # $a68=a68
0x040007b 0x230fffff addi $29,$29,00 90: add $a69,$a69,$zero # $a69=a69
0x040007c 0x230fffff addi $29,$29,00 91: add $a70,$a70,$zero # $a70=a70
0x040007d 0x230fffff addi $29,$29,00 92: add $a71,$a71,$zero # $a71=a71
0x040007e 0x230fffff addi $29,$29,00 93: add $a72,$a72,$zero # $a72=a72
0x040007f 0x230fffff addi $29,$29,00 94: add $a73,$a73,$zero # $a73=a73
0x0400080 0x230fffff addi $29,$29,00 95: add $a74,$a74,$zero # $a74=a74
0x0400081 0x230fffff addi $29,$29,00 96: add $a75,$a75,$zero # $a75=a75
0x0400082 0x230fffff addi $29,$29,00 97: add $a76,$a76,$zero # $a76=a76
0x0400083 0x230fffff addi $29,$29,00 98: add $a77,$a77,$zero # $a77=a77
0x0400084 0x230fffff addi $29,$29,00 99: add $a78,$a78,$zero # $a78=a78
0x0400085 0x230fffff addi $29,$29,00 100: add $a79,$a79,$zero # $a79=a79
0x0400086 0x230fffff addi $29,$29,00 101: add $a80,$a80,$zero # $a80=a80
0x0400087 0x230fffff addi $29,$29,00 102: add $a81,$a81,$zero # $a81=a81
0x0400088 0x230fffff addi $29,$29,00 103: add $a82,$a82,$zero # $a82=a82
0x0400089 0x230fffff addi $29,$29,00 104: add $a83,$a83,$zero # $a83=a83
0x040008a 0x230fffff addi $29,$29,00 105: add $a84,$a84,$zero # $a84=a84
0x040008b 0x230fffff addi $29,$29,00 106: add $a85,$a85,$zero # $a85=a85
0x040008c 0x230fffff addi $29,$29,00 107: add $a86,$a86,$zero # $a86=a86
0x040008d 0x230fffff addi $29,$29,00 108: add $a87,$a87,$zero # $a87=a87
0x040008e 0x230fffff addi $29,$29,00 109: add $a88,$a88,$zero # $a88=a88
0x040008f 0x230fffff addi $29,$29,00 110: add $a89,$a89,$zero # $a89=a89
0x0400090 0x230fffff addi $29,$29,00 111: add $a90,$a90,$zero # $a90=a90
0x0400091 0x230fffff addi $29,$29,00 112: add $a91,$a91,$zero # $a91=a91
0x0400092 0x230fffff addi $29,$29,00 113: add $a92,$a92,$zero # $a92=a92
0x0400093 0x230fffff addi $29,$29,00 114: add $a93,$a93,$zero # $a93=a93
0x0400094 0x230fffff addi $29,$29,00 115: add $a94,$a94,$zero # $a94=a94
0x0400095 0x230fffff addi $29,$29,00 116: add $a95,$a95,$zero # $a95=a95
0x0400096 0x230fffff addi $29,$29,00 117: add $a96,$a96,$zero # $a96=a96
0x0400097 0x230fffff addi $29,$29,00 118: add $a97,$a97,$zero # $a97=a97
0x0400098 0x230fffff addi $29,$29,00 119: add $a98,$a98,$zero # $a98=a98
0x0400099 0x230fffff addi $29,$29,00 120: add $a99,$a99,$zero # $a99=a99
0x040009a 0x230fffff addi $29,$29,00 121: add $a100,$a100,$zero # $a100=a100
0x040009b 0x230fffff addi $29,$29,00 122: add $a101,$a101,$zero # $a101=a101
0x040009c 0x230fffff addi $29,$29,00 123: add $a102,$a102,$zero # $a102=a102
0x040009d 0x230fffff addi $29,$29,00 124: add $a103,$a103,$zero # $a103=a103
0x040009e 0x230fffff addi $29,$29,00 125: add $a104,$a104,$zero # $a104=a104
0x040009f 0x230fffff addi $29,$29,00 126: add $a105,$a105,$zero # $a105=a105
0x04000a0 0x230fffff addi $29,$29,00 127: add $a106,$a106,$zero # $a106=a106
0x04000a1 0x230fffff addi $29,$29,00 128: add $a107,$a107,$zero # $a107=a107
0x04000a2 0x230fffff addi $29,$29,00 129: add $a108,$a108,$zero # $a108=a108
0x04000a3 0x230fffff addi $29,$29,00 130: add $a109,$a109,$zero # $a109=a109
0x04000a4 0x230fffff addi $29,$29,00 131: add $a110,$a110,$zero # $a110=a110
0x04000a5 0x230fffff addi $29,$29,00 132: add $a111,$a111,$zero # $a111=a111
0x04000a6 0x230fffff addi $29,$29,00 133: add $a112,$a112,$zero # $a112=a112
0x04000a7 0x230fffff addi $29,$29,00 134: add $a113,$a113,$zero # $a113=a113
0x04000a8 0x230fffff addi $29,$29,00 135: add $a114,$a114,$zero # $a114=a114
0x04000a9 0x230fffff addi $29,$29,00 136: add $a115,$a115,$zero # $a115=a115
0x04000aa 0x230fffff addi $29,$29,00 137: add $a116,$a116,$zero # $a116=a116
0x04000ab 0x230fffff addi $29,$29,00 138: add $a117,$a117,$zero # $a117=a117
0x04000ac 0x230fffff addi $29,$29,00 139: add $a118,$a118,$zero # $a118=a118
0x04000ad 0x230fffff addi $29,$29,00 140: add $a119,$a119,$zero # $a119=a119
0x04000ae 0x230fffff addi $29,$29,00 141: add $a120,$a120,$zero # $a120=a120
0x04000af 0x230fffff addi $29,$29,00 142: add $a121,$a121,$zero # $a121=a121
0x04000b0 0x230fffff addi $29,$29,00 143: add $a122,$a122,$zero # $a122=a122
0x04000b1 0x230fffff addi $29,$29,00 144: add $a123,$a123,$zero # $a123=a123
0x04000b2 0x230fffff addi $29,$29,00 145: add $a124,$a124,$zero # $a124=a124
0x04000b3 0x230fffff addi $29,$29,00 146: add $a125,$a125,$zero # $a125=a125
0x04000b4 0x230fffff addi $29,$29,00 147: add $a126,$a126,$zero # $a126=a126
0x04000b5 0x230fffff addi $29,$29,00 148: add $a127,$a127,$zero # $a127=a127
0x04000b6 0x230fffff addi $29,$29,00 149: add $a128,$a128,$zero # $a128=a128
0x04000b7 0x230fffff addi $29,$29,00 150: add $a129,$a129,$zero # $a129=a129
0x04000b8 0x230fffff addi $29,$29,00 151: add $a130,$a130,$zero # $a130=a130
0x04000b9 0x230fffff addi $29,$29,00 152: add $a131,$a131,$zero # $a131=a131
0x04000ba 0x230fffff addi $29,$29,00 153: add $a132,$a132,$zero # $a132=a132
0x04000bb 0x230fffff addi $29,$29,00 154: add $a133,$a133,$zero # $a133=a133
0x04000bc 0x230fffff addi $29,$29,00 155: add $a134,$a134,$zero # $a134=a134
0x04000bd 0x230fffff addi $29,$29,00 156: add $a135,$a135,$zero # $a135=a135
0x04000be 0x230fffff addi $29,$29,00 157: add $a136,$a136,$zero # $a136=a136
0x04000bf 0x230fffff addi $29,$29,00 158: add $a137,$a137,$zero # $a137=a137
0x04000c0 0x230fffff addi $29,$29,00 159: add $a138,$a138,$zero # $a138=a138
0x04000c1 0x230fffff addi $29,$29,00 160: add $a139,$a139,$zero # $a139=a139
0x04000c2 0x230fffff addi $29,$29,00 161: add $a140,$a140,$zero # $a140=a140
0x04000c3 0x230fffff addi $29,$29,00 162: add $a141,$a141,$zero # $a141=a141
0x04000c4 0x230fffff addi $29,$29,00 163: add $a142,$a142,$zero # $a142=a142
0x04000c5 0x230fffff addi $29,$29,00 164: add $a143,$a143,$zero # $a143=a143
0x04000c6 0x230fffff addi $29,$29,00 165: add $a144,$a144,$zero # $a144=a144
0x04000c7 0x230fffff addi $29,$29,00 166: add $a145,$a145,$zero # $a145=a145
0x04000c8 0x230fffff addi $29,$29,00 167: add $a146,$a146,$zero # $a146=a146
0x04000c9 0x230fffff addi $29,$29,00 168: add $a147,$a147,$zero # $a147=a147
0x04000ca 0x230fffff addi $29,$29,00 169: add $a148,$a148,$zero # $a148=a148
0x04000cb 0x230fffff addi $29,$29,00 170: add $a149,$a149,$zero # $a149=a149
0x04000cc 0x230fffff addi $29,$29,00 171: add $a150,$a150,$zero # $a150=a150
0x04000cd 0x230fffff addi $29,$29,00 172: add $a151,$a151,$zero # $a151=a151
0x04000ce 0x230fffff addi $29,$29,00 173: add $a152,$a152,$zero # $a152=a152
0x04000cf 0x230fffff addi $29,$29,00 174: add $a153,$a153,$zero # $a153=a153
0x04000d0 0x230fffff addi $29,$29,00 175: add $a154,$a154,$zero # $a154=a154
0x04000d1 0x230fffff addi $29,$29,00 176: add $a155,$a155,$zero # $a155=a155
0x04000d2 0x230fffff addi $29,$29,00 177: add $a156,$a156,$zero # $a156=a156
0x04000d3 0x230fffff addi $29,$29,00 178: add $a157,$a157,$zero # $a157=a157
0x04000d4 0x230fffff addi $29,$29,00 179: add $a158,$a158,$zero # $a158=a158
0x04000d5 0x230fffff addi $29,$29,00 180: add $a159,$a159,$zero # $a159=a159
0x04000d6 0x230fffff addi $29,$29,00 181: add $a160,$a160,$zero # $a160=a160
0x04000d7 0x230fffff addi $29,$29,00 182: add $a161,$a161,$zero # $a161=a161
0x04000d8 0x230fffff addi $29,$29,00 183: add $a162,$a162,$zero # $a162=a162
0x04000d9 0x230fffff addi $29,$29,00 184: add $a163,$a163,$zero # $a163=a163
0x04000da 0x230fffff addi $29,$29,00 185: add $a164,$a164,$zero # $a164=a164
0x04000db 0x230fffff addi $29,$29,00 186: add $a165,$a165,$zero # $a165=a165
0x04000dc 0x230fffff addi $29,$29,00 187: add $a166,$a166,$zero # $a166=a166
0x04000dd 0x230fffff addi $29,$29,00 188: add $a167,$a167,$zero # $a167=a167
0x04000de 0x230fffff addi $29,$29,00 189: add $a168,$a168,$zero # $a168=a168
0x04000df 0x230fffff addi $29,$29,00 190: add $a169,$a169,$zero # $a169=a169
0x04000e0 0x230fffff addi $29,$29,00 191: add $a170,$a170,$zero # $a170=a170
0x04000e1 0x230fffff addi $29,$29,00 192: add $a171,$a171,$zero # $a171=a171
0x04000e2 0x230fffff addi $29,$29,00 193: add $a172,$a172,$zero # $a172=a172
0x04000e3 0x230fffff addi $29,$29,00 194: add $a173,$a173,$zero # $a173=a173
0x04000e4 0x230fffff addi $29,$29,00 195: add $a174,$a174,$zero # $a174=a174
0x04000e5 0x230fffff addi $29,$29,00 196: add $a175,$a175,$zero # $a175=a175
0x04000e6 0x230fffff addi $29,$29,00 197: add $a176,$a176,$zero # $a176=a176
0x04000e7 0x230fffff addi $29,$29,00 198: add $a177,$a177,$zero # $a177=a177
0x04000e8 0x230fffff addi $29,$29,00 199: add $a178,$a178,$zero # $a178=a178
0x04000e9 0x230fffff addi $29,$29,00 200: add $a179,$a179,$zero # $a179=a179
0x04000ea 0x230fffff addi $29,$29,00 201: add $a180,$a180,$zero # $a180=a180
0x04000eb 0x230fffff addi $29,$29,00 202: add $a181,$a181,$zero # $a181=a181
0x04000ec 0x230fffff addi $29,$29,00 203: add $a182,$a182,$zero # $a182=a182
0x04000ed 0x230fffff addi $29,$29,00 204: add $a183,$a183,$zero # $a183=a183
0x04000ee 0x230fffff addi $29,$29,00 205: add $a184,$a184,$zero # $a184=a184
0x04000ef 0x230fffff addi $29,$29,00 206: add $a185,$a185,$zero # $a185=a185
0x04000f0 0x230fffff addi $29,$29,00 207: add $a186,$a186,$zero # $a186=a186
0x04000f1 0x230fffff addi $29,$29,00 208: add $a187,$a187,$zero # $a187=a187
0x04000f2 0x230fffff addi $29,$29,00 209: add $a188,$a188,$zero # $a188=a188
0x04000f3 0x230fffff addi $29,$29,00 210: add $a189,$a189,$zero # $a189=a189
0x04000f4 0x230fffff addi $29,$29,00 211: add $a190,$a190,$zero # $a190=a190
0x04000f5 0x230fffff addi $29,$29,00 212: add $a191,$a191,$zero # $a191=a191
0x04000f6 0x230fffff addi $29,$29,00 213: add $a192,$a192,$zero # $a192=a192
0x04000f7 0x230fffff addi $29,$29,00 214: add $a193,$a193,$zero # $a193=a193
0x04000f8 0x230fffff addi $29,$29,00 215: add $a194,$a194,$zero # $a194=a194
0x04000f9 0x230fffff addi $29,$29,00 216: add $a195,$a195,$zero # $a195=a195
0x04000fa 0x230fffff addi $29,$29,00 217: add $a196,$a196,$zero # $a196=a196
0x04000fb 0x230fffff addi $29,$29,00 218: add $a197,$a197,$zero # $a197=a197
0x04000fc 0x230fffff addi $29,$29,00 219: add $a198,$a198,$zero # $a198=a198
0x04000fd 0x230fffff addi $29,$29,00 220: add $a199,$a199,$zero # $a199=a199
0x04000fe 0x230fffff addi $29,$29,00 221: add $a200,$a200,$zero # $a200=a200
0x04000ff 0x230fffff addi $29,$29,00 222:
```

In figure 24, we can see that it is storing 2nd argument and the register, \$sp, value is 0x7fffefd8. The offset is 0 and value is 0x00000000c.

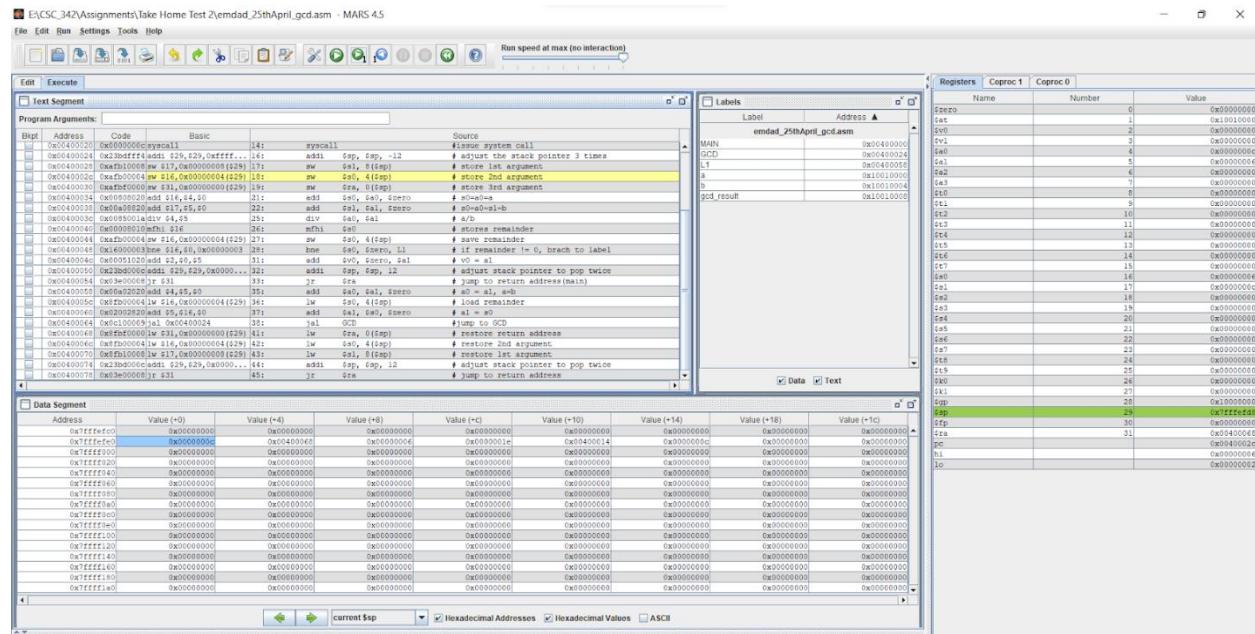


Figure 24: storing 2nd argument

In figure 25, we can see that it is storing 3rd argument and the register, \$sp, value is 0x7fffefd8. The offset is 0 and value is 0x00000000c.

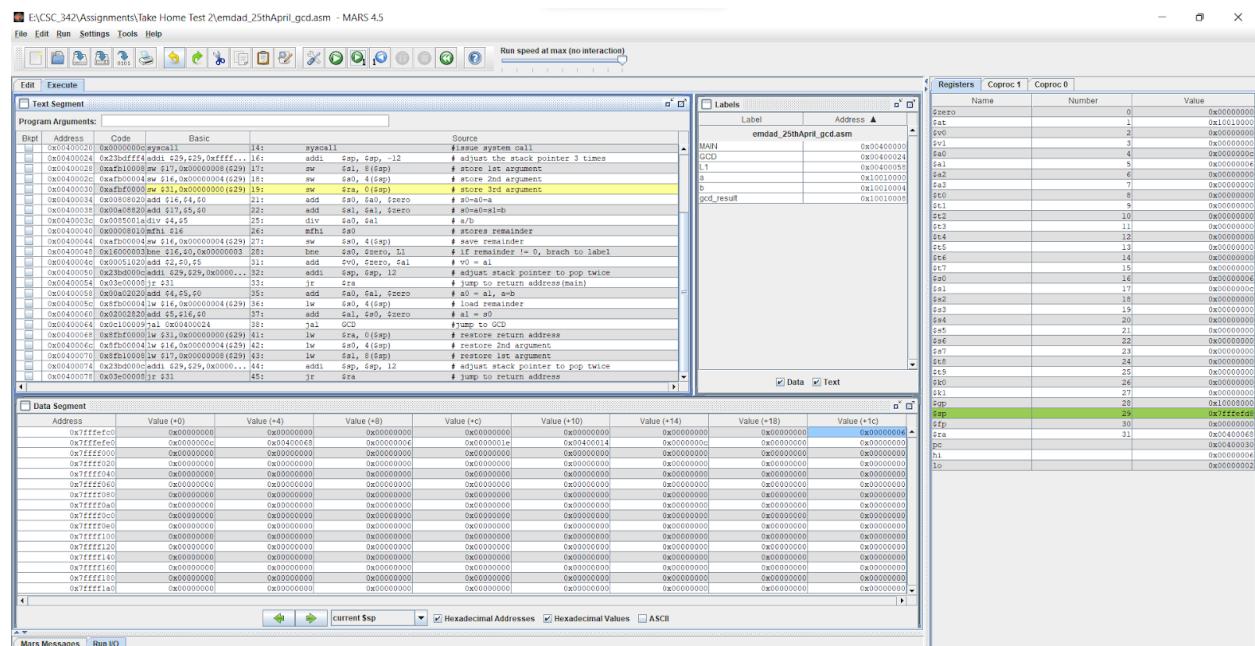


Figure 25: storing 3rd argument

In figure 26, we can see that line 41 is restoring the return address. The register moved to \$ra and value is 0x00400068 and the memory offset is +1c and value is 0x00000000.

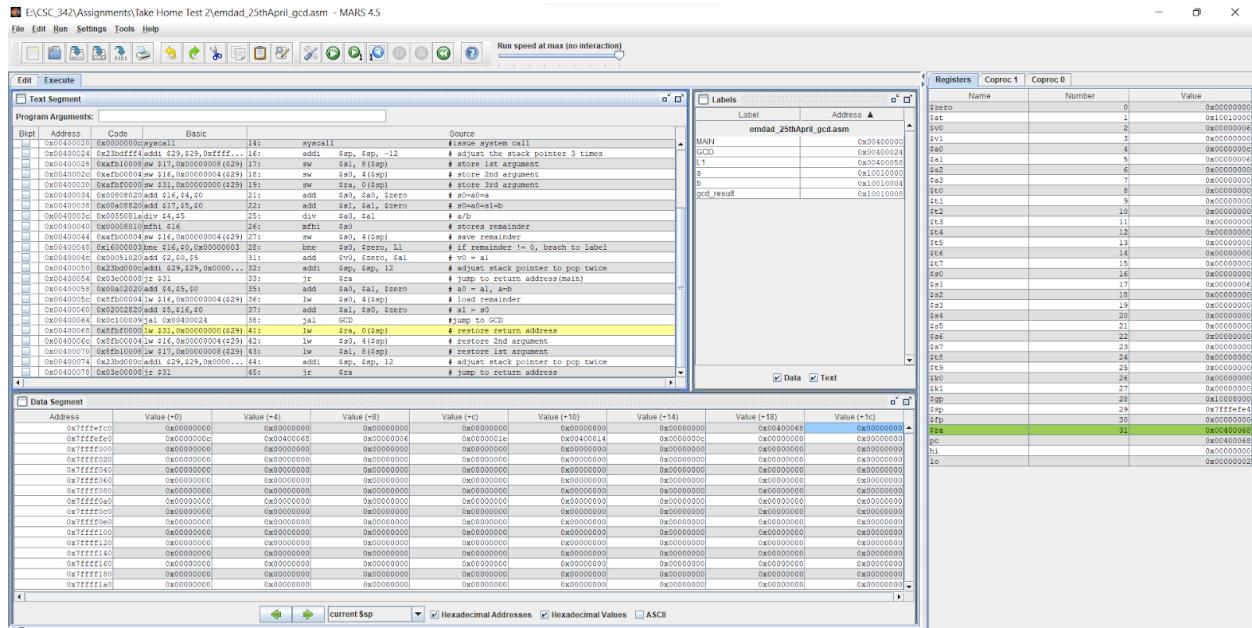


Figure 26: restore return address

In figure 27, we can see that line 42 is restoring 2nd argument.

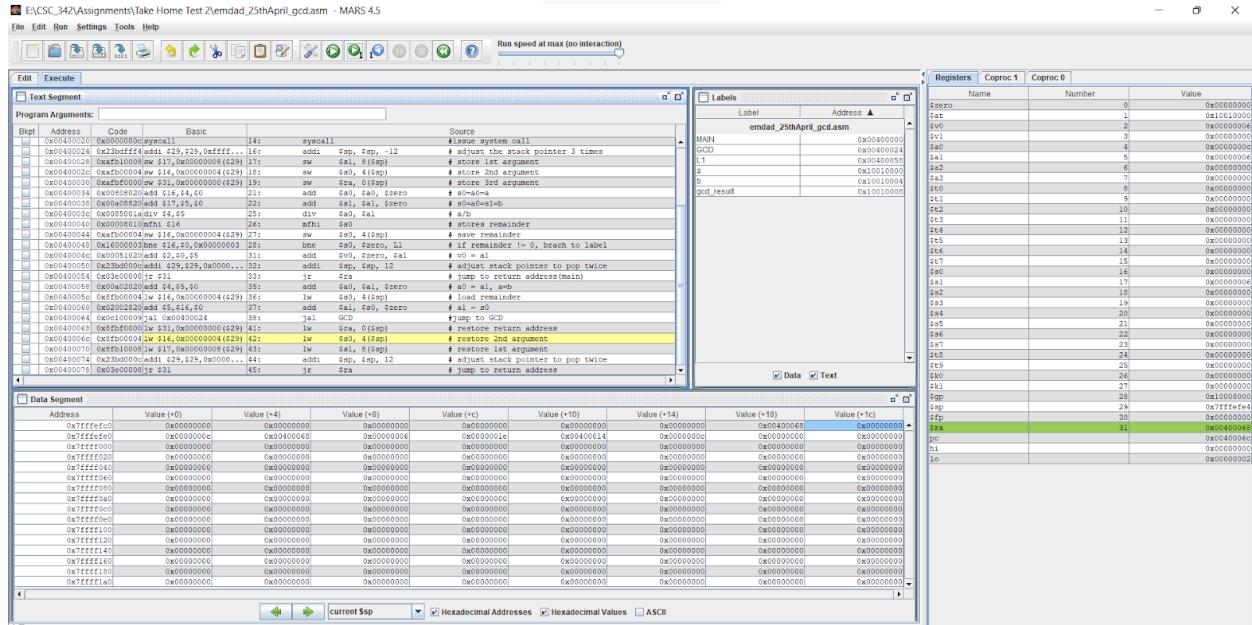


Figure 27: restoring 2nd argument

In figure 28, we can see that line 43 is restoring 1st argument.

Figure 28: restoring 1st argument

In figure 29, we can see that line 43 is adjusting the stack pointer and pop registers.

Figure 29: adjusting the stack pointer

In figure 30, we can see that line 45 jumps back to return address and register is \$sp and value of its is 0x7ffffeff.

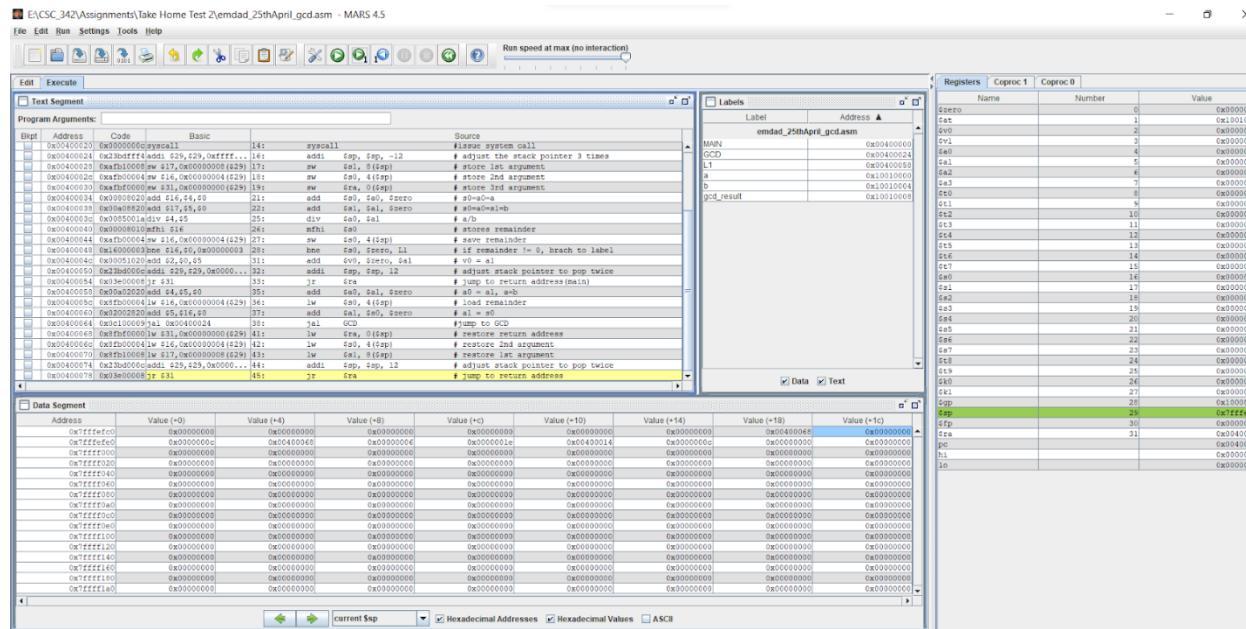


Figure 30: jumps back to return address

In figure 31, we can see that the syscall is closed and process ends. The value of \$v0 is 0x000000a. The offset +8 has the value of 6 which is our gcd.

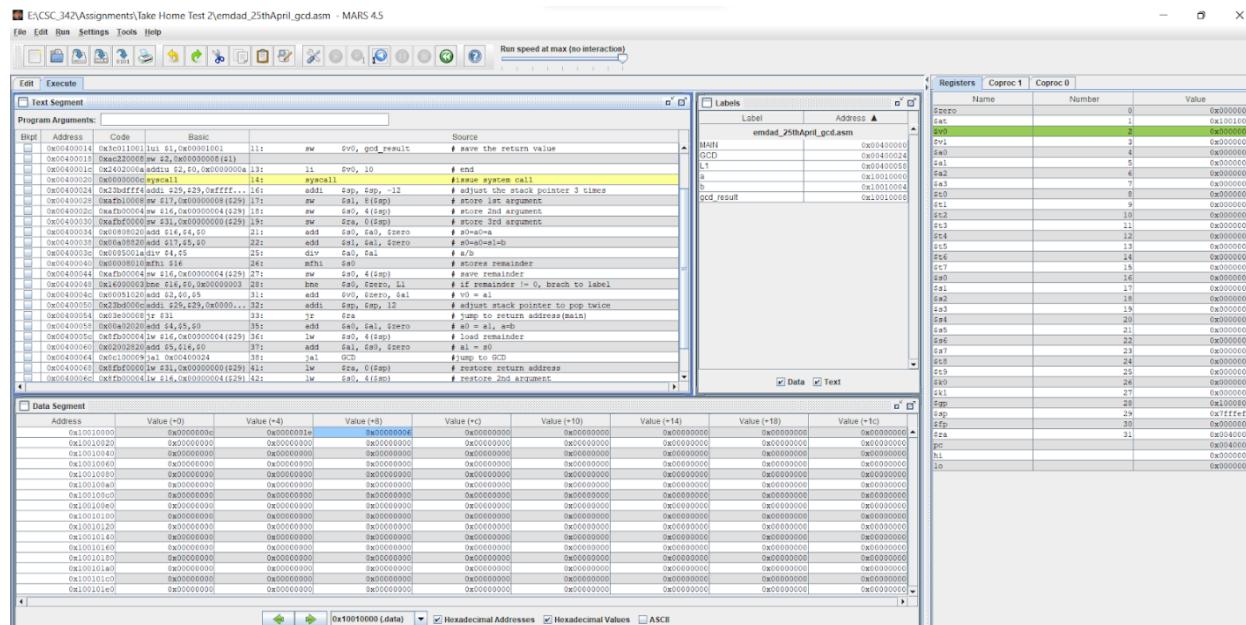


Figure 31: returns gcd

PART II: Visual Studio 2019 (Intel x86):

In figure 32, we can see the project directory for “emdad_25thApril_gcd”.

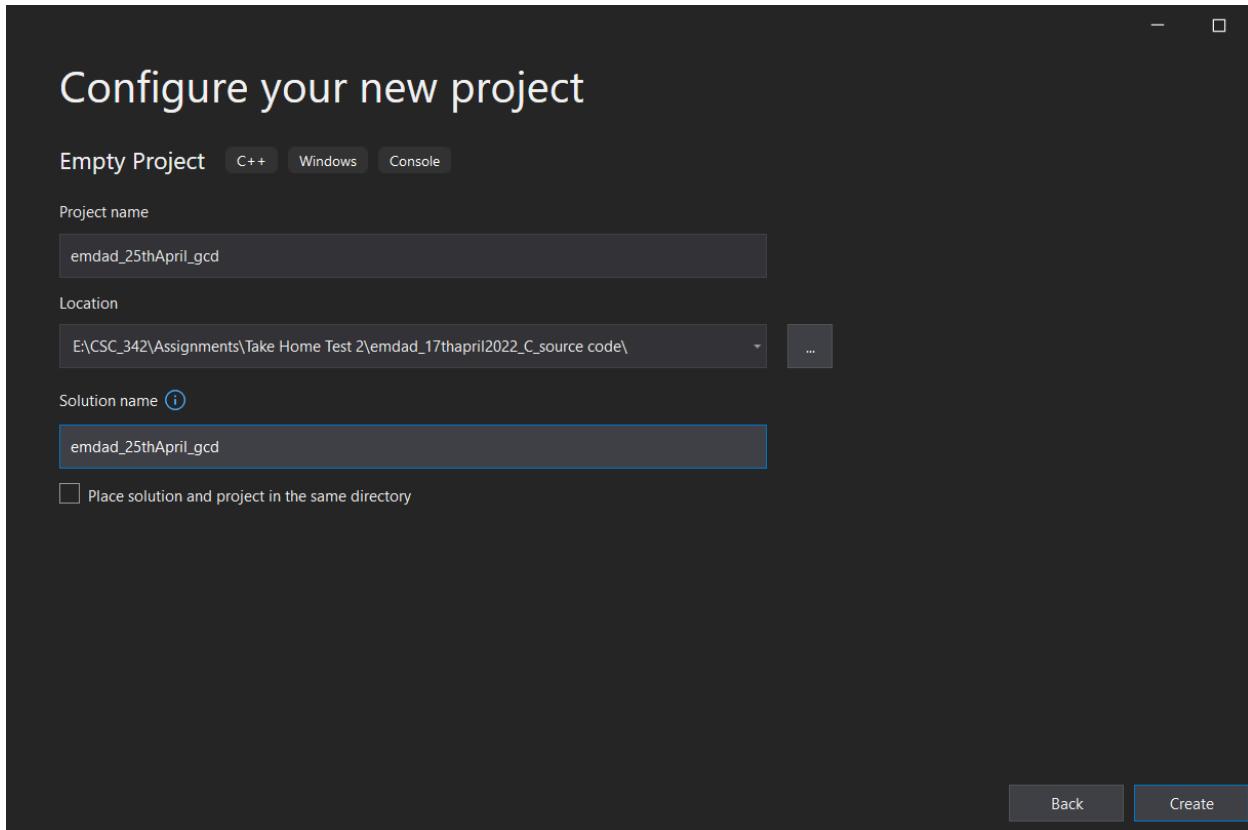


Figure 32: project directory for “emdad_25thApril_gcd”

In figure 33, we can see that I am adding .c file to the project which will perform gcd(a,b).

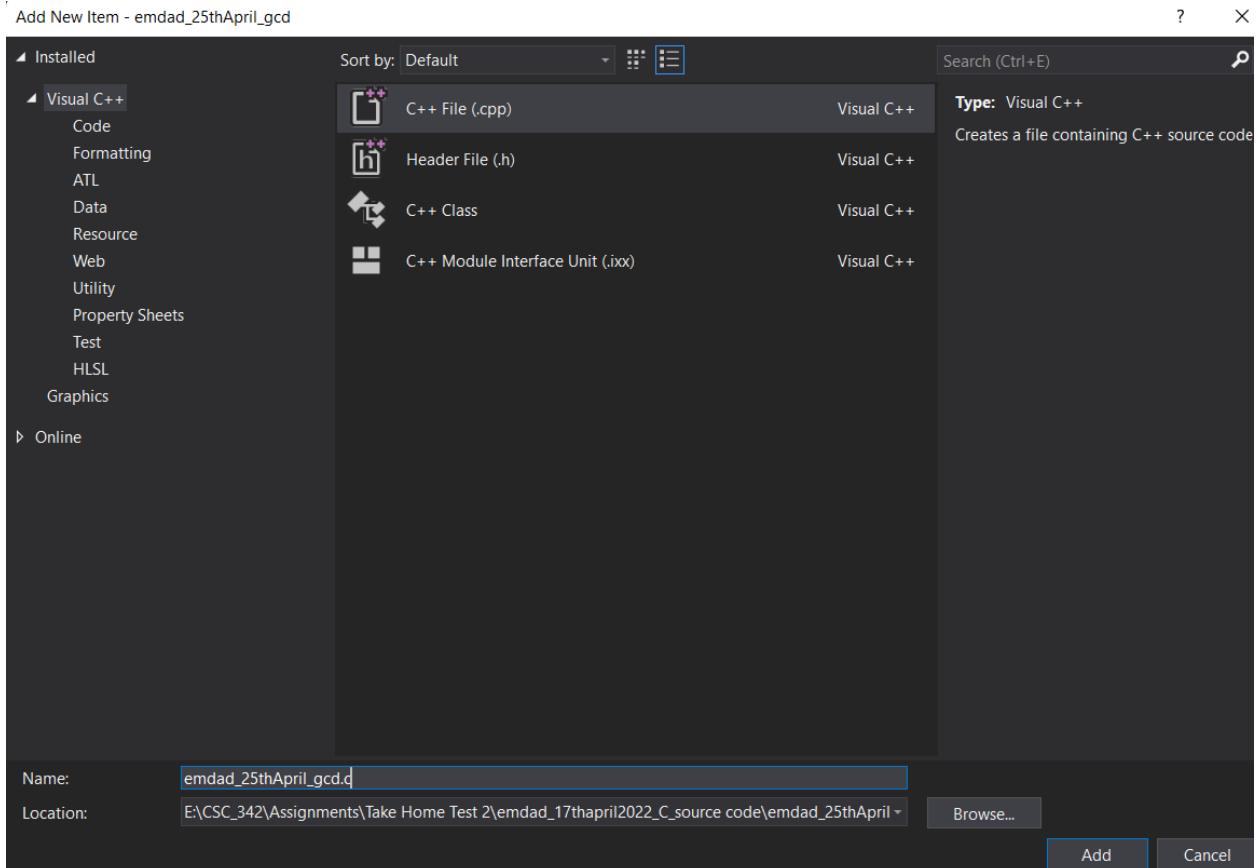


Figure 33: creating .c file

In figure 34, we can see that the code to perform gcd and it built successfully.

```

emdad_25thApril_gcd.c  ✘
emdad_25thApril_gcd
1 int gcd(int a, int b) {
2     if (a == 0) {
3         return b;
4     }
5     return gcd(b % a, a);
6 }
7
8 void main() {
9     int a = 12;
10    int b = 30;
11    int gcd_result = gcd(a, b);
12 }

100 %  No issues found
Output
Show output from: Debug
'emdad_25thApril_gcd.exe' (Win32): Loaded 'E:\CSC_342\Assignments\Take Home Test 2\emdad_17thApril2022_C_source code\emdad_25thApril\Debug\emdad_25thApril_gcd.exe'. 
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\ntdll.dll'.
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\kernel32.dll'.
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\KernelBase.dll'.
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\apphelp.dll'.
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\vcrunrtime140d.dll'.
'emdad_25thApril_gcd.exe' (Win32): Loaded 'C:\Windows\SysWOW64\ucrtbased.dll'.
The thread 0xb24 has exited with code 0 (0x0).
The program '[7708] emdad_25thApril_gcd.exe' has exited with code 0 (0x0).
|

```

Figure 34: gcd code and built successfully.

In figure 35, we can see the registers window at the top and disassembly at the left side and memory address on the right side.

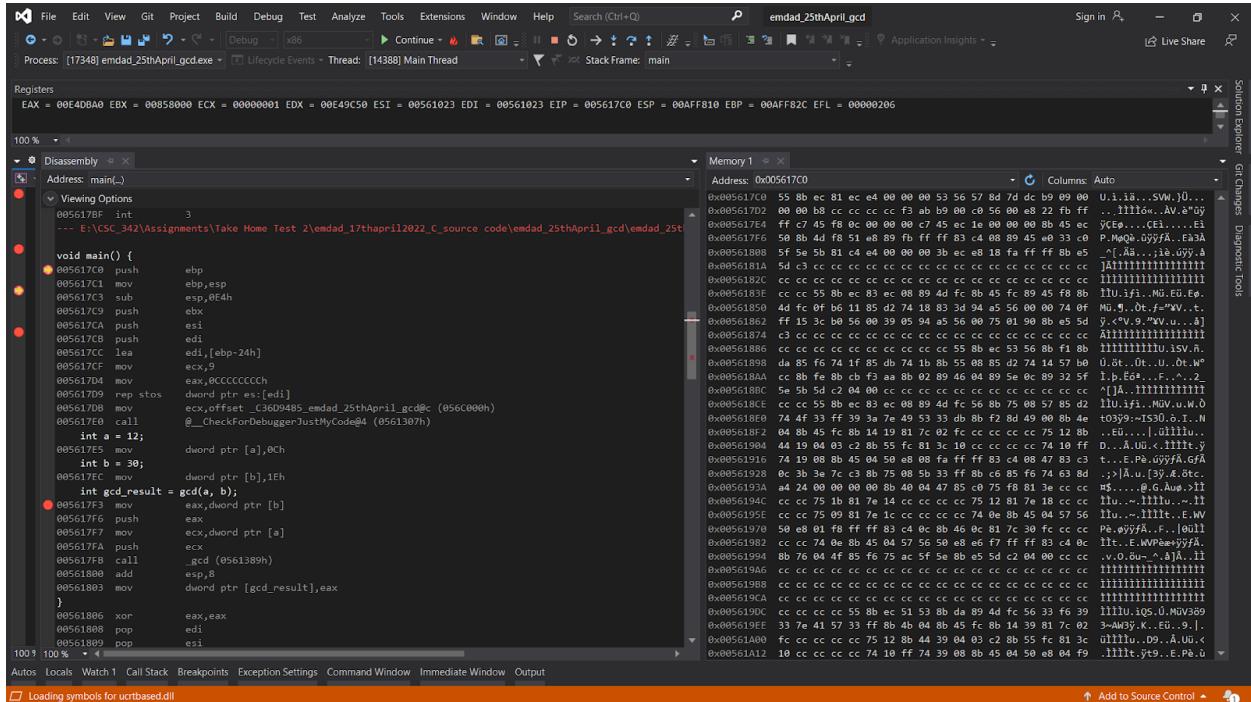


Figure 35: registers, disassembly, and memory address window

In figure 36, we can see the registers, EAX = 0000001E, ECX = 0000000C, EIP = 00C31750, ESP = 0077F928. The memory address 0x00561840 and 0x005618D0 has same EIP and EAX.

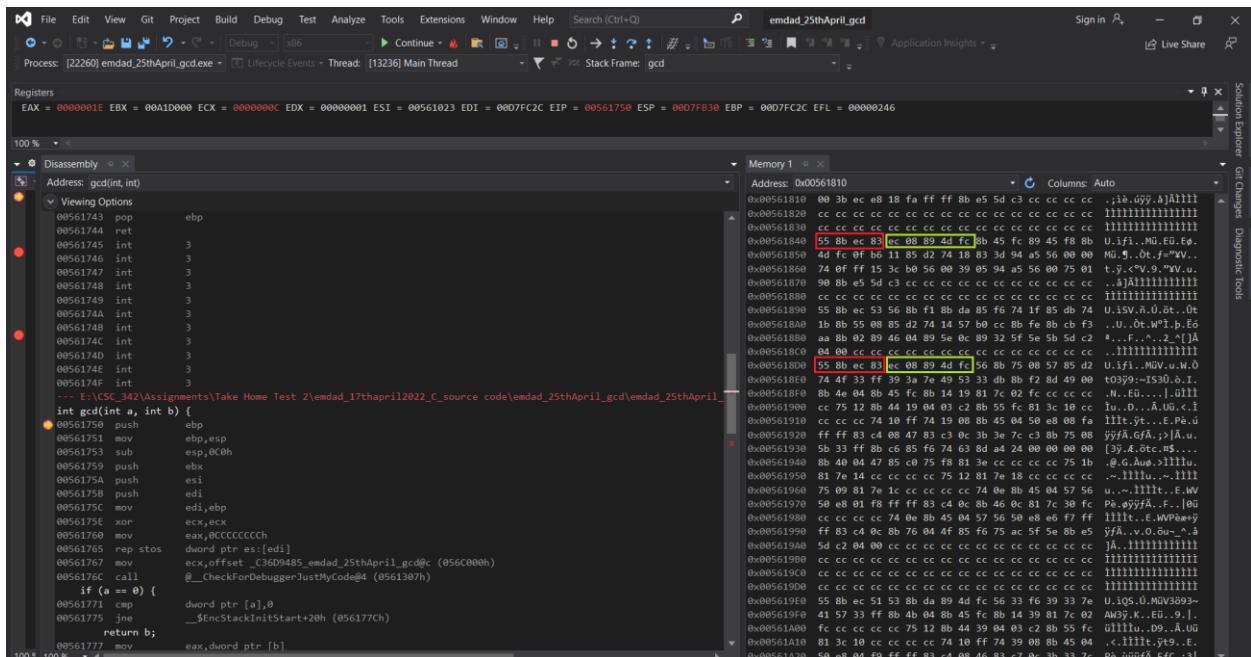


Figure 36: breakpoints (Part 1)

In figure 37, we can see that EAX = 00000002 EBX = 00408000 ECX = 00C3C000 EDX = 00000006 ESI = 00C31023 EDI = 006FFD84 EIP = 00C31750 ESP = 006FFCAC EBP = 006FFD84 EFL = 00000206.

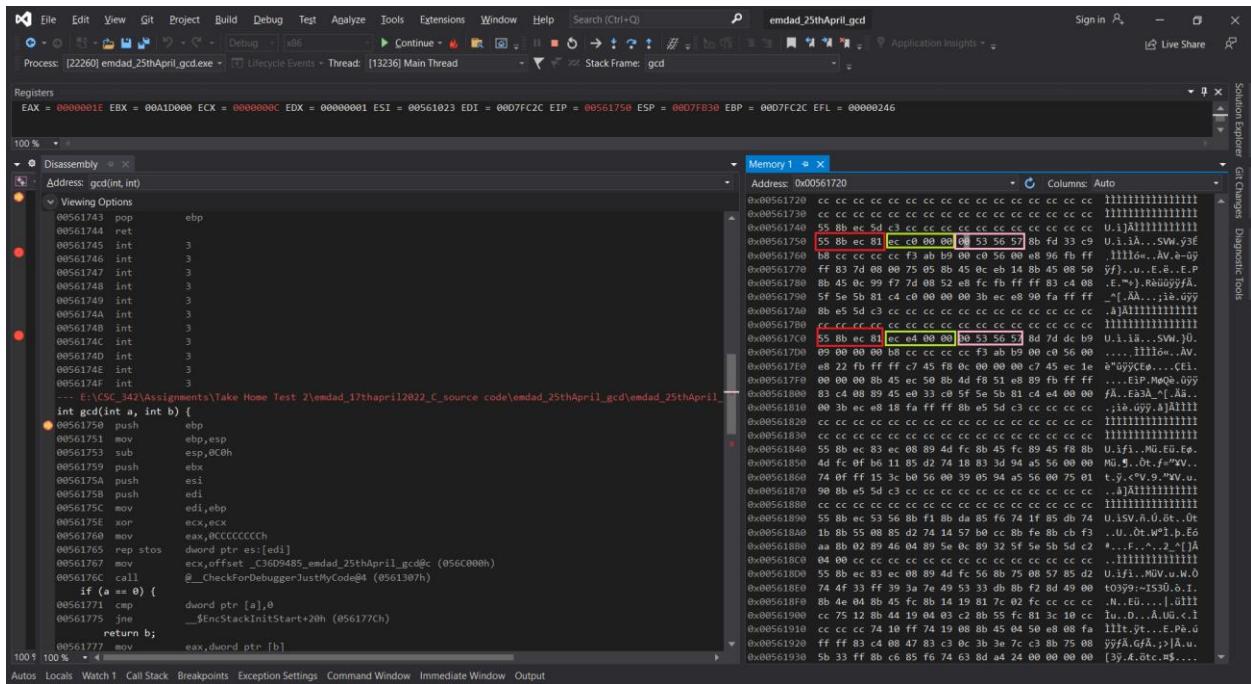


Figure 37: breakpoints (Part 2)

In figure 38, we can see that EAX = 00C3C000 EBX = 00408000 ECX = 00C3C000 EDX = 00000001 ESI = 00C31023 EDI = 006FFCA8 EIP = 00C3177C ESP = 006FFBDC EBP = 006FFCA8 EFL = 00000206.

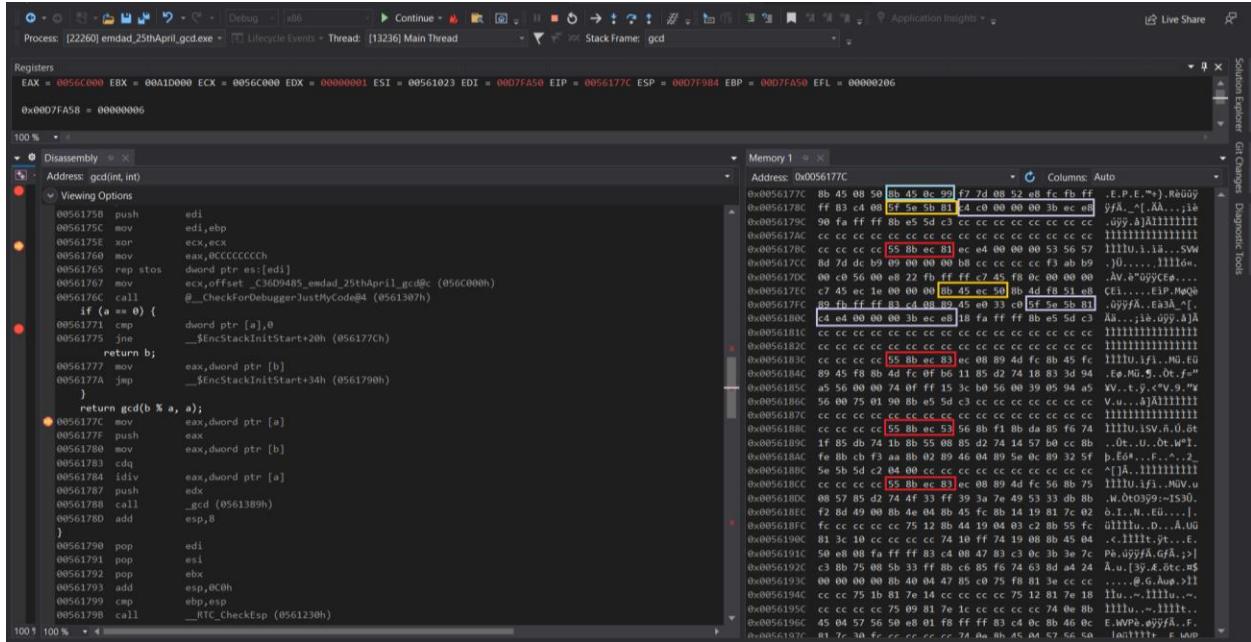


Figure 38: breakpoints (Part 3)

PART III: x86 64 ISA, Linux 64-bit:

In figure 39, we can see the source code for emdad_25thApril_gcd.c in linux editor.

```

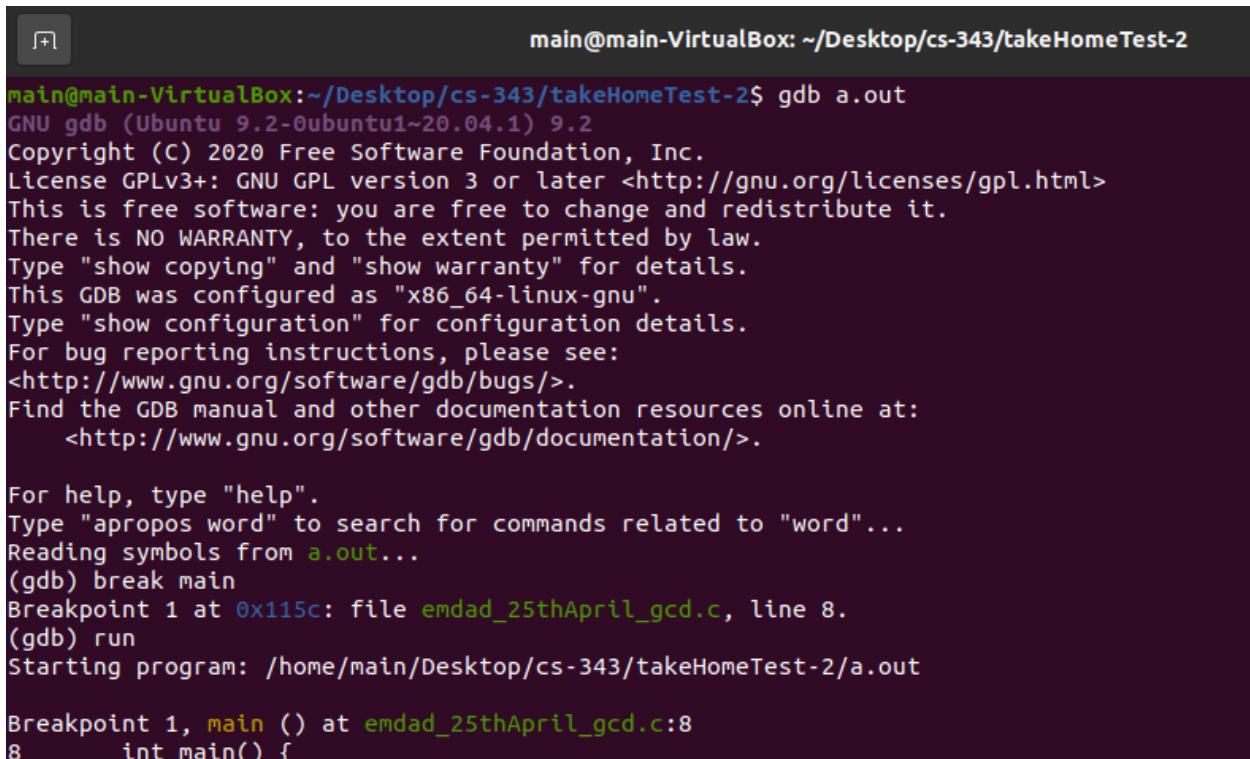
Documents Open emdad_25thApril_gcd.c ~/Desktop/cs-343/takeHomeTest2 Save
emdad_25thApril_gcd.c x
1 int gcd(int a, int b) {
2     if (a == 0) {
3         return b;
4     }
5     return gcd(b % a, a);
6 }
7
8 int main() {
9     int a = 12;
10    int b = 30;
11    int gcd_result = gcd(a, b);
12 }

```

Figure 39: source code for emdad_25thApril_gcd.c

The main function started at line 10 and then there are some variables initialized. We called the gcd function in the main function and printed the value of it. The gcd function is a recursion function which call itself and keep running until a == 0. The gcd for 12 and 30 is 6.

In figure 40, we can see that we compiled the code and used gdb a.out to run gdb. Then we did break main to put a breakpoint on the main function. Then, I did run to run the gbd and select-frame is 0.



```

main@main-VirtualBox:~/Desktop/cs-343/takeHomeTest-2$ gdb a.out
GNU gdb (Ubuntu 9.2-0ubuntu1~20.04.1) 9.2
Copyright (C) 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
Type "show copying" and "show warranty" for details.
This GDB was configured as "x86_64-linux-gnu".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
  <http://www.gnu.org/software/gdb/documentation/>.

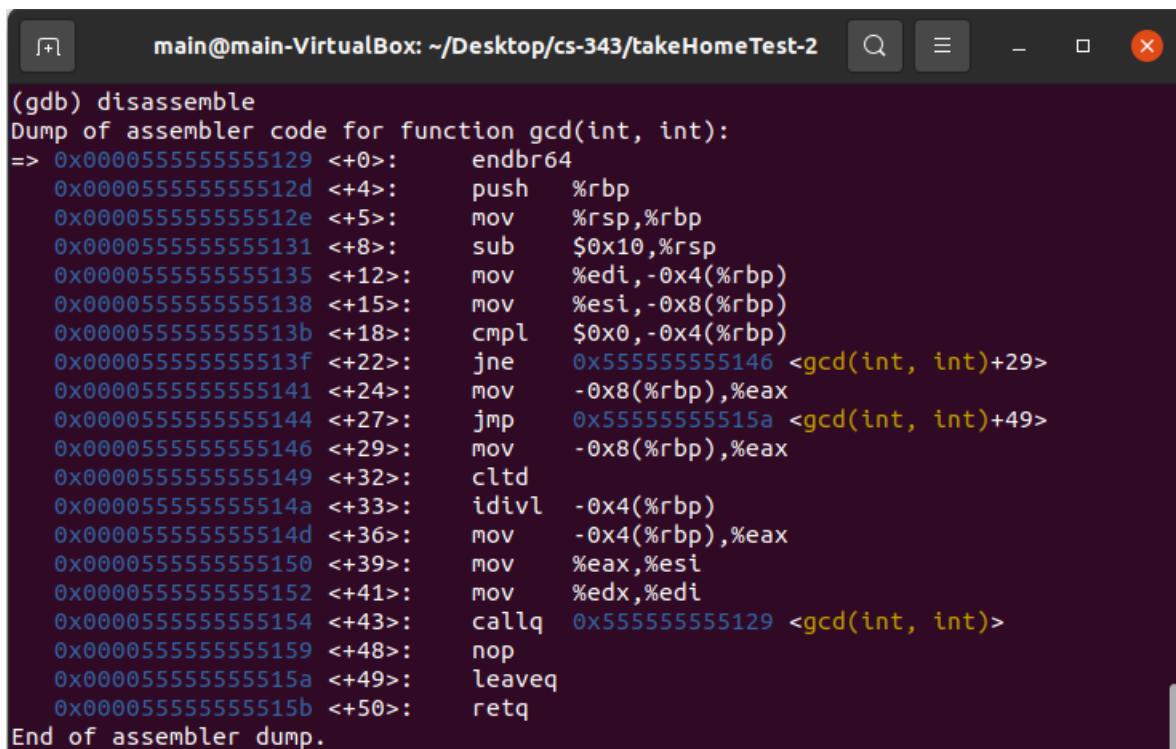
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from a.out...
(gdb) break main
Breakpoint 1 at 0x115c: file emdad_25thApril_gcd.c, line 8.
(gdb) run
Starting program: /home/main/Desktop/cs-343/takeHomeTest-2/a.out

Breakpoint 1, main () at emdad_25thApril_gcd.c:8
8      int main() {

```

Figure 40: code compilation and run gdb

In figure 41, we can see the disassembly code.



```

(gdb) disassemble
Dump of assembler code for function gcd(int, int):
=> 0x000055555555129 <+0>:    endbr64
 0x00005555555512d <+4>:    push   %rbp
 0x00005555555512e <+5>:    mov    %rsp,%rbp
 0x000055555555131 <+8>:    sub    $0x10,%rsp
 0x000055555555135 <+12>:   mov    %edi,-0x4(%rbp)
 0x000055555555138 <+15>:   mov    %esi,-0x8(%rbp)
 0x00005555555513b <+18>:   cmpl   $0x0,-0x4(%rbp)
 0x00005555555513f <+22>:   jne    0x55555555146 <gcd(int, int)+29>
 0x000055555555141 <+24>:   mov    -0x8(%rbp),%eax
 0x000055555555144 <+27>:   jmp    0x5555555515a <gcd(int, int)+49>
 0x000055555555146 <+29>:   mov    -0x8(%rbp),%eax
 0x000055555555149 <+32>:   cltd
 0x00005555555514a <+33>:   idivl  -0x4(%rbp)
 0x00005555555514d <+36>:   mov    -0x4(%rbp),%eax
 0x000055555555150 <+39>:   mov    %eax,%esi
 0x000055555555152 <+41>:   mov    %edx,%edi
 0x000055555555154 <+43>:   callq  0x55555555129 <gcd(int, int)>
 0x000055555555159 <+48>:   nop
 0x00005555555515a <+49>:   leaveq 
 0x00005555555515b <+50>:   retq

End of assembler dump.

```

Figure 41: disassembly code.

After running the disassemble code, we get the assembler code of function main. The rsp is the stack pointer and the rbp stores that pointer. Callq calling the function gcd and then storing the address of the new generated value. The mov instruction stores all local values in registers.

```
(gdb) nexti 3
0x0000555555555131      1      int gcd(int a, int b) {
```

Figure 42: nexti command

In figure 42, we can see the nexti 3 online and the address is 0x0000555555555131.

```
(gdb) printf "rbp:%x\nrsp:%x\n", $rbp,$rsp
rbp:fffffdf40
rsp:fffffdf40
```

Figure 43: print rbp and rsp

In figure 43, we can see the print value of rbp and rsp by using the command:

```
printf "rbp:%x\nrsp:%x\n", $rbp,$rsp
```

```
(gdb) x/12xw $rsp
0x7fffffffdf40: 0xfffffdf60      0x00007fff      0x55555185      0x00005555
0x7fffffffdf50: 0xfffffe050      0x0000000c      0x00000001e     0x000000000
0x7fffffffdf60: 0x00000000      0x00000000      0xf7de70b3      0x00007fff
```

Figure 44: print RBP, return address, argument

In figure 44, we can see that the saved RBP which is 0xffffe050 and 0x0000000c and return address is 0x00000001e and 0x000000000.

In figure 42, we can see the info registers of the code.

```
(gdb) info registers
rax          0x555555555515c    93824992235868
rbx          0x5555555555190    93824992235920
rcx          0x5555555555190    93824992235920
rdx          0x7fffffff068     140737488347240
rsi          0x7fffffff058     140737488347224
rdi          0x1              1
rbp          0x0              0x0
rsp          0x7fffffffdf68   0x7fffffffdf68
r8           0x0              0
r9           0x7fffff7fe0d50   140737354009936
r10          0x0              0
r11          0x0              0
r12          0x5555555555040   93824992235584
r13          0x7fffffff050     140737488347216
r14          0x0              0
r15          0x0              0
rip          0x555555555515c    0x555555555515c <main()>
eflags        0x246          [ PF ZF IF ]
cs            0x33           51
ss            0x2b           43
ds            0x0             0
es            0x0             0
fs            0x0             0
gs            0x0             0
(gdb)
```

Figure 45: *info registers* of the code.

The RBP register is the base pointer. The add instruction calculates the addition of registers edx and eax. These two registers contain local data. Subcommands are subtracted. Memory address of a local static variable.

Explanation:

The MIPS MARS simulator is a 32-bit processor, and each instruction it executes is also 32-bit long. MIPS processors have registers such as zeros, stack pointers, frame pointers, and program counters. A stack frame is a data frame (that is, an area of memory) that is created (that is, an area of memory). In the (allocated) stack. A new stack frame is created each time the function is called. The new frame is initialized by pushing the old base pointer onto the stack. The new base pointer. It is initialized and allocates space (by subtraction) for the required data from the stack pointer. Next, all arguments are pushed onto the stack (using a negative offset using the local base pointer) variable). Finally, the return address (that is, the address after the function call) is shifted up. This process in the stack repeats for each function call before proceeding to the next function call. For new stack frames. The behavior of stack frames may vary from program to program and best ways to understand stack frames is to use recursion.

In contrast to MIPS, Intel has one operand in a register. The second operand can be placed in memory. Transfer the data using the "mov" instruction. Intel implements a loop with jne and jmp. Intel function calls are made using the call instruction, which sets an instruction pointer to an instruction that requests a jump to the function. The return address is stored on the stack for later use when the function call completes. Static variables are stored in memory away from the stack. Local variables are assigned positions in the stack, and the values are initially loaded directly into memory rather than into registers.

I ran a GDB disassembler using Oracle VirtualMachine. Ubuntu Linux instructions. The data is stored in memory in little endian notation. The result of the instruction is stored in the register on the right, unlike Intel, which was stored in the register on the left. Moving data between registers and memory is done using mov. This is the same as Intel, copying the data from the left operand to the right operand.

Time-Comparison:

In figure 46, we can see the time comparison table for input vs execution time.

A	B	Time(s)
12	30	0.000152
100	50	0.000153
1000	500	0.000154
10000	50000	0.000155

Figure 46: time comparison table

In figure 47, we can see the graph execution for the table in chart.

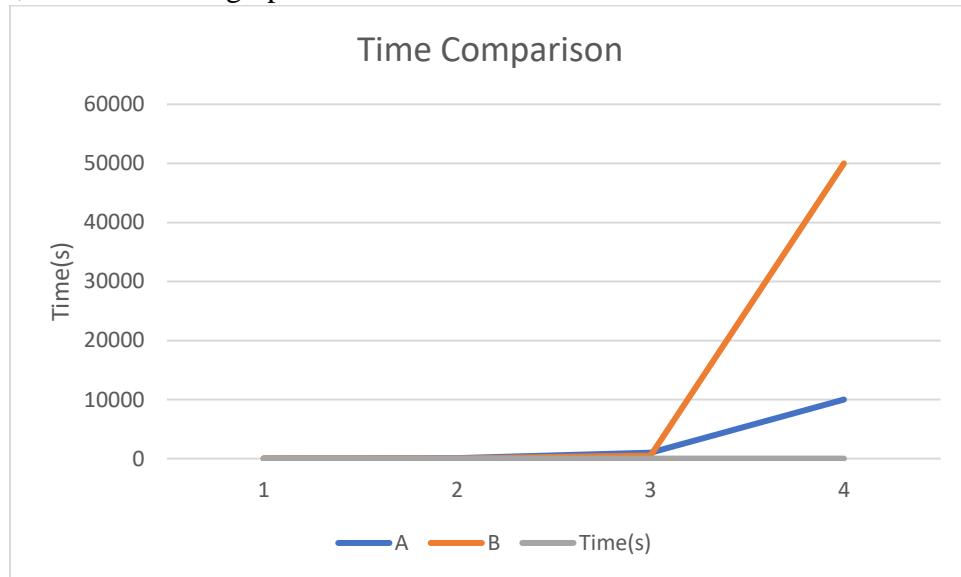


Figure 47: graph for GCD execution time

Conclusion:

This take-home test taught me how to demonstrate recursive calls and stack frames on MIPS instructions set architecture, Intel x86 ISA using Windows MS 32-bit compiler and debugger and a Intel X86_64 bit ISA processor running Linux, 64 bit gcc and gdb. Further, I learned and understood about registers, memory, and disassembly and how to debug code overall.