

Computer Architecture Lab

LAB 1: MACHINE LANGUAGE INSTRUCTIONS, WARMING UP WITH
SIMULATOR, HIGH LEVEL-TO-LOW LEVEL LANGUAGE CONVERSION

Code used for Assignments 1 and 2

```
# Name : Jeffrey Huang

# Exercise 1 is used in assignments 1 and 2

.data 0x10000000
.text 0x00400000
.globl main

main:
    addi $10, $0, 8          # add immediate : $10 = $0 + 8
    li $11, 6                # Place (load immediate) 6 into register $11.
    # calculate the $11 -th power of $10
    add $8, $0, $10          # add : $8 = $0 + $10
    li $12, 2                # load 2 into $12

powerloop:
    bgt $12, $11, powerexit  # go to powerexit if $12 > $11
    sub $11, $11, 1          # subtract : $11 = $11 - 1
    mul $10, $10, $8         # multiply : $10 = $10 * $8
    j powerloop              # jump to powerloop

powerexit:
    # power operation loop is over
    # Is the result in $10 correct? The result in $10 is hexadecimal
    li $2, 10                # load 10 into $2
    syscall                  # End program
```

Assignment 1:

Instructions	op	rs	rt	rd	shamt	funct
add \$8, \$0, \$10	0000 00	ss sss	t tttt	dddd d	000 00	10 0000
	op	rs	rt	Constant		
addi \$10, \$0, 8	0010 00	ss sss	t tttt	iiii iiii iiii iiii		

Assignment 2:

Comments are show in the code above.

Code used for Assignment 3

```
# Jeffrey Huang

# Exercise2 is used in assignment 3

# $a0 : Size
# $a1 : i
# $t1 : $a1 * $t2
# $t2 : 4

.text 0x00400000
.align 2
.globl main

main:
    lw $a0, Size          # load the size of array into $a0, using lw
    addi $a0, $a0, -1     # init index j with size - 1
    li $a1, 0             # init index i = 0
    li $t2, 4             # t2 contains constant 4, init t2

loop:
    mul $t1, $a1, $t2      # t1 gets i*4
    mul $t4, $a0, $t2      # t4 gets j*4
    lw $a3, Array1($t1)    # a3 = Array[i]
    mul $a3, $a3, 2        # double the value a3 = a3 * 2
    sw $a3, Array1($t4)    # store a3 to Array1[j]
    addi $a1, $a1, 1       # i = i + 1
    sub $a0, $a0, 1        # j = j - 1
    bge $a1, $a0, END     # j = j - 1
    j loop

END:
    li $v0, 10            # syscall code 10 is for exit
    syscall               # make the syscall

.data 0x10000000
.align 2
Array1: .word 2 5 6 7 12 16 25 27
Size: .word 8

# The reason why the index j is (size - 1) but not size is
# because the first index of the array is 0 and not 1. So
# the index counts from 0 to 7 which is the total number of
# 8 counts.
```

Assignment 3:

1. Missing instructions according to the comments can be seen above.

PC	= 400054	R12 [t4]	= 10
EPC	= 0	R13 [t5]	= 0
Cause	= 0	R14 [t6]	= 0
BadVAddr	= 0	R15 [t7]	= 0
Status	= 3000ff10	R16 [s0]	= 0
		R17 [s1]	= 0
HI	= 0	R18 [s2]	= 0
LO	= e	R19 [s3]	= 0
		R20 [s4]	= 0
R0 [r0]	= 0	R21 [s5]	= 0
R1 [at]	= 0	R22 [s6]	= 0
R2 [v0]	= a	R23 [s7]	= 0
R3 [v1]	= 0	R24 [t8]	= 0
R4 [a0]	= 3	R25 [t9]	= 0
R5 [a1]	= 4	R26 [k0]	= 0
R6 [a2]	= 7ffff444	R27 [k1]	= 0
R7 [a3]	= e	R28 [gp]	= 10008000
R8 [t0]	= 0	R29 [sp]	= 7ffff42c
R9 [t1]	= c	R30 [s8]	= 0
R10 [t2]	= 4	R31 [ra]	= 0
R11 [t3]	= 0		

- 2.

User data segment [10000000]..[10040000]
[10000000] 00000002 00000005 00000006 00000007
[10000010] 0000000e 0000000c 0000000a 00000004
[10000020] 00000008 00000000 00000000 00000000
[10000030]..[1003ffff] 00000000

User Stack [7ffff42c]..[80000000]
[7ffff42c] 00000004
[7ffff430] 7ffff530 7ffff519 7ffff50c 7ffff500 0
[7ffff440] 00000000 7fffffe1 7fffffb2 7fffff91
[7ffff450] 7fffff5a 7fffff1e 7fffffed 7ffffed8 Z
[7ffff460] 7ffffeb4 7ffffe8a 7ffffe59 7ffffe31 Y 1
[7ffff470] 7ffffe1d 7ffffde8 7ffffddb 7ffffdbd
[7ffff480] 7ffffd8b 7ffffd75 7ffffd5e 7ffffd50 u ^ P
[7ffff490] 7ffff90a 7ffff8cc 7ffff8b1 7ffff894
[7ffff4a0] 7ffff84c 7ffff83a 7ffff822 7ffff807 L "
[7ffff4b0] 7ffff7e3 7ffff7ba 7ffff79c 7ffff75b [. . . .
[7ffff4c0] 7ffff744 7ffff730 7ffff721 7ffff70b D !
[7ffff4d0] 7ffff6e1 7ffff6b8 7ffff6a5 7ffff683
[7ffff4e0] 7ffff66c 7ffff649 7ffff5f7 7ffff5a5 1 I
[7ffff4f0] 7ffff553 7ffff541 00000000 00000000 S A
[7ffff500] 2e6214c 2e327845 006d7361 68637241 L a b / E x 2 . a s m . A r c h
[7ffff510] 63657469 65727574 61754800 442f676e i t e c t u r e . H u a n g / D
[7ffff520] 746b7365 432f706f 75706d6f 00726574 e s k t o p / C o m p u t e r .
[7ffff530] 552f3a43 73726573 66654a2f 79657266 C : / U s e r s / J e f f r e y
[7ffff540] 66697700 3d726964 575c3a43 4f444e49 . w i n d i r = C : \ W I N D O
[7ffff550] 56005357 30343153 4e4d4f43 4c4f4f54 W S . V S 1 4 0 C O M N T O O L
[7ffff560] 3a433d53 6f72505c 6d617267 6c694620 S = C : \ P r o g r a m F i l
[7ffff570] 28207365 29363878 63694d5c 6f736f72 e s (x 8 6) \ M i c r o s o
[7ffff580] 56207466 61757369 7453206c 6f696475 f t V i s u a l S t u d i o
[7ffff590] 2e343120 6f435c30 6e6f6d6d 6f545c37 1 4 . 0 \ C o m m o n 7 \ T o
[7ffff5a0] 5c736c6f 31535600 4f433032 4f544e4d o l s \ . V S 1 2 0 C O M N T O
[7ffff5b0] 3d534c4f 505c3a43 72676f72 46206d61 O L S = C : \ P r o g r a m F
[7ffff5c0] 73656c69 38782820 4d5c2936 6f726369 i l e s (x 8 6) \ M i c r o
[7ffff5d0] 74666f73 73695620 206c6175 64757453 s o f t V i s u a l S t u d
[7ffff5e0] 31206f69 5c302e32 6d6d6f43 5c376e6f i o 1 2 . 0 \ C o m m o n 7 \
[7ffff5f0] 6c6f6f54 56005c73 30313153 4e4d4f43 T o o l s \ . V S 1 1 0 C O M N
[7ffff600] 4c4f4f54 3a433d53 6f72505c 6d617267 T O O L S = C : \ P r o g r a m
[7ffff610] 6c694620 28207365 29363878 63694d5c F i l e s (x 8 6) \ M i c
[7ffff620] 6f736f72 56207466 61757369 7453206c r o s o f t V i s u a l S t
[7ffff630] 6f696475 2e313120 6f435c30 6e6f6d6d u d i o 1 1 . 0 \ C o m m o n
[7ffff640] 2e3b5441 3b444d43 5342562e 42562e3b A T ; . C M D ; . V B S ; . V B
[7ffff650] 4a2e3b45 4a2e3b53 2e3b4553 3b465357 E ; . J S ; . J S E ; . W S F ;
[7ffff660] 4853572e 534d2e3b 61500043 433d6874 . W S H ; . M S C . P a t h = C
[7ffff670] 72505c3a 6172676f 7461446d 724f5c61 : \ P r o g r a m D a t a \ O r
[7ffff680] 656c6361 76614a5c 616a5c61 61706176 a c l e \ J a v a \ j a v a p a
[7ffff690] 433b6874 72505c3a 6172676f 6946206d t h ; C : \ P r o g r a m F i
[7ffff700] 2073656c 36387828 6e495c29 5c6c6574 l e s (x 8 6) \ I n t e l \
[7ffff710] 534c4369 696c4320 5c746e65 5c3a433b i C L S C l i e n t \ ; C : \
[7ffff720] 676f7250 206d6172 656c6946 6e495c73 P r o g r a m F i l e s \ I n
[7ffff730] 5c6c6574 534c4369 696c4320 5c746e65 t e l \ i C L S C l i e n t \
[7ffff740] 5c3a433b 444e4957 5c53574f 74737973 ; C : \ W I N D O W S \ s y s t
[7ffff750] 32336d65 5c3a433b 444e4957 3b53574f e m 3 2 ; C : \ W I N D O W S ;
[7ffff760] 575c3a43 4f444e49 535c5357 65747379 C : \ W I N D O W S \ S y s t e
[7ffff770] 5c32336d 6d656257 5c3a433b 444e4957 m 3 2 \ W b e m ; C : \ W I N D
[7ffff780] 5c53574f 74737953 32336d65 6e69575c O W S \ S y s t e m 3 2 \ W i n
[7ffff790] 73776f64 65776f50 65685372 765c6c6c d o w s P o w e r S h e l l \ v
[7ffff800] 5c302e31 5c3a433b 676f7250 206d6172 1 . 0 \ ; C : \ P r o g r a m
[7ffff810] 656c6946 6e495c73 5c6c6574 65746e49 F i l e s \ I n t e l \ I n t e
[7ffff820] 2952286c 6e614d20 6d656761 20746e65 l (R) M a n a g e m e n t
[7ffff830] 69676e45 4320656e 6f706d6f 746e656e E n g i n e C o m p o n e n t
[7ffff840] 41445c73 3a433b4c 6f72505c 6d617267 s \ D A L ; C : \ P r o g r a m
[7ffff850] 6c694620 28207365 29363878 746e495c F i l e s (x 8 6) \ I n t
[7ffff860] 495c6c65 6c65746e 20295228 616e614d e l \ I n t e l (R) M a n a
[7ffff870] 656d6567 4520746e 6e69676e 6f432065 g e m e n t E n g i n e C o
[7ffff880] 6e6f706d 73746e65 4c41445c 5c3a433b m p o n e n t s \ D A L ; C : \
[7ffff890] 676f7250 206d6172 656c6946 6e495c73 P r o g r a m F i l e s \ I n
[7ffff900] 5c6c6574 65746e49 2952286c 6e614d20 t e l \ I n t e l (R) M a n
[7ffff910] 6d656761 20746e65 69676e45 4320656e a g e m e n t E n g i n e C
[7ffff920] 6f706d6f 746e656e 50495c73 3a433b54 m p o n e n t s \ I P T ; C :
[7ffff930] 6f72505c 6d617267 6c694620 28207365 \ P r o g r a m F i l e s (
[7ffff940] 29363878 746e495c 495c6c65 6c65746e x 8 6) \ I n t e l \ I n t e l
[7ffff950] 20295228 616e614d 656d6567 4520746e (R) M a n a g e m e n t E
[7ffff960] 6e69676e 6f432065 6e6f706d 73746e65 n g i n e C o m p o n e n t s
[7ffff970] 5450495c 5c3a433b 676f7250 206d6172 \ I P T ; C : \ P r o g r a m
[7ffff980] 656c6946 78282073 5c293638 4449564e F i l e s (x 8 6) \ N V I D
[7ffff990] 43204149 6f70726f 69746172 505c6e6f I A C o r p o r a t i o n \ P
[7ffffa00] 58737968 6d6f435c 3b6e6f6d 505c3a43 h y s X \ C o m m o n ; C : \ P
[7ffffa10] 72676f72 46206d61 73656c69 63694d5c r o g r a m F i l e s \ M i c
[7ffffa20] 6f736f72 53207466 53204c51 65767265 r o s o f t S Q L S e r v e
[7ffffa30] 32315c72 6f545c30 5c736c6f 6e6e6942 r \ 1 2 0 \ T o o l s \ B i n n
[7ffffa40] 3a433b5c 6f72505c 6d617267 6c694620 \ ; C : \ P r o g r a m F i l
[7ffffa50] 28207365 29363878 646f6e5c 5c736a65 e s (x 8 6) \ n o d e j s \

3.

[7ffff640] 6f545c37 5c736c6f 45535500 4f525052 7\Tools\USERPRO
[7ffff650] 454c4946 5c3a433d 72657355 654a5c73 FILE=C:\Users\Jef
[7ffff660] 65726666 75482079 00676e61 52455355 ffreY Huang.USER
[7ffff670] 454d414e 66654a3d 79657266 61754820 NAME=Jeffrey Hua
[7ffff680] 5500676e 44524553 49414d4f 4f525f4e ng.USERDOMAIN=RO
[7ffff690] 4e494d41 4f525047 454c4946 66654a3d AMINGPROFILE=Jef
[7ffff6a0] 43502d66 45535500 4d4f4452 3d4e4941 f-PC.USERDOMAIN=
[7ffff6b0] 6666654a 0043502d 3d504d54 555c3a43 Jeff-PC.TMP=C:\U
[7ffff6c0] 73726573 46454a5c 7e455246 70415c31 sers\JEFFRE~1\Ap
[7ffff6d0] 74614470 6f4c5c61 5c6c6163 706d6554 pData\Local\Temp
[7ffff6e0] 4d455400 3a433d50 6573555c 4a5c7372 .TMP=C:\Users\J
[7ffff6f0] 52464645 5c317e45 44707041 5c617461 EFFRE~1\AppData\
[7ffff700] 61636f4c 65545c6c 5300706d 65747379 Local\Temp.Syste
[7ffff710] 6f6f526d 3a433d74 4e49575c 53574f44 mRoot=C:\WINDOWS
[7ffff720] 73795300 446d6574 65766972 003a433d .SystemDrive=C:
[7ffff730] 53534553 4e4e4f49 3d454d41 736e6f43 SESSIONNAME=Cons
[7ffff740] 00656c6f 4c425550 433d4349 73555c3a ole.PUBLIC=C:\Us
[7ffff750] 5c737265 6c627550 50006369 646f4d53 ers\Public.PSMod
[7ffff760] 50656c75 3d687461 575c3a43 4f444e49 ulePath=C:\WINDO
[7ffff770] 735c5357 65747379 5c32336d 646e6957 WS\system32\Wind
[7ffff780] 5073776f 7265776f 6c656853 31765c6c owsPowerShell\v1
[7ffff790] 4d5c302e 6c75646f 005c7365 676f7250 .0\Modules\Prog
[7ffff7a0] 576d6172 32333436 5c3a433d 676f7250 ramW6432=C:\Prog
[7ffff7b0] 206d6172 656c6946 72500073 6172676f ramFiles.Progra
[7ffff7c0] 6c69466d 78287365 3d293638 505c3a43 mFiles(x86)=C:\P
[7ffff7d0] 72676f72 46206d61 73656c69 38782820 rogramFiles(x8
[7ffff7e0] 50002936 72676f72 69466d61 3d73656c 6).ProgramFiles=
[7ffff7f0] 505c3a43 72676f72 46206d61 73656c69 C:\ProgramFiles
[7ffff800] 38782820 50002936 72676f72 61446d61 (x86).ProgramDat
[7ffff810] 433d6174 72505c3a 6172676f 7461446d ta=C:\ProgramDat
[7ffff820] 52500061 5345434f 5f524f53 49564552 a.PROCESSOR_REVI
[7ffff830] 4e4f4953 3064333d 52500034 5345434f SION=3d04.PROCES
[7ffff840] 5f524f53 4556454c 00363d4c 434f5250 SOR_LEVEL=6.PROC
[7ffff850] 4f535345 44495f52 49544e45 52454946 ESSOR_IDENTIFIER
[7ffff860] 746e493d 34366c65 6d614620 20796c69 =Intel64 Family
[7ffff870] 6f4d2036 206c6564 53203136 70706574 6 Model 61 Stepp
[7ffff880] 20676e69 47202c34 69756e65 6e49656e ing 4, GenuineIn
[7ffff890] 006c6574 434f5250 4f535345 52415f52 tel.PROCESSOR_AR
[7ffff8a0] 54494843 34365745 413d3233 3436444d CHITW6432=AMD64
[7ffff8b0] 4f525000 53534543 415f524f 49484352 .PROCESSOR_ARCHI
[7ffff8c0] 54434554 3d455255 00363878 48544150 TECTURE=x86.PATH
[7ffff8d0] 3d545845 4d4f432e 58452e3b 422e3b45 EXT=.COM;.EXE;.B
[7ffff8e0] 5c3a433b 444e4957 5c53574f 74737973 ;C:\WINDOWS\sys
[7ffff8f0] 32336d65 5c3a433b 444e4957 3b53574f em32;C:\WINDOWS;
[7ffff900] 575c3a43 4f444e49 535c5357 65747379 C:\WINDOWS\Syste
[7ffff910] 5c32336d 6d656257 5c3a433b 444e4957 m32\Wbem;C:\WIND
[7ffff920] 5c53574f 74737953 32336d65 6e69575c OWS\System32\Win
[7ffff930] 73776f64 65776f50 65685372 765c6c6c dowsPowerShell\v
[7ffff940] 5c302e31 5c3a433b 676f7250 206d6172 1.0\;C:\Program
[7ffff950] 656c6946 78282073 5c293638 70796b53 Files(x86)\Skyp
[7ffff960] 68505c65 5c656e6f 5c3a433b 4143724f e\Phone\ \ORCA
[7ffff970] 724f5c44 5f444143 362e3631 74694c5f D\ORCAD_16.6_Lit
[7ffff980] 704f5c65 63416e65 73736563 6e69625c e\OpenAccess\bin
[7ffff990] 6e69775c 6f5c3233 433b7470 724f5c3a \win32\opt;C:\OR
[7ffff9a0] 5c444143 4143724f 36315f44 4c5f362e CAD\ORCAD_16.6_L
[7ffff9b0] 5c657469 6c6f6f74 63705c73 69625c62 ite\tools\pcb\bi
[7ffff9c0] 3a433b6e 43724f5c 4f5c4441 44414372 n;C:\ORCAD\ORCAD
[7ffff9d0] 2e36315f 694c5f36 745c6574 736c6f6f _16.6_Lite\tools
[7ffff9e0] 7465665c 6e69625c 5c3a433b 4143724f \fet\bin;C:\ORCA
[7ffff9f0] 724f5c44 5f444143 362e3631 74694c5f D\ORCAD_16.6_Lit
[7ffffa00] 6f745c65 5c736c6f 74706143 3b657275 e\tools\Capture;
[7ffffa10] 4f5c3a43 44414372 43724f5c 315f4441 C:\ORCAD\ORCAD_1
[7ffffa20] 5f362e36 6574694c 6f6f745c 505c736c 6.6_Lite\tools\P
[7ffffa30] 63697053 3a433b65 43724f5c 4f5c4441 Spice;C:\ORCAD\O
[7ffffa40] 44414372 2e36315f 694c5f36 745c6574 RCAD_16.6_Lite\to
[7ffffa50] 736c6f6f 6570735c 72746363 69625c61 ools\spec\tra\bi
[7ffffa60] 3a433b6e 43724f5c 4f5c4441 44414372 n;C:\ORCAD\ORCAD
[7ffffa70] 2e36315f 694c5f36 745c6574 736c6f6f _16.6_Lite\tools
[7ffffa80] 6e69625c 5c3a433b 72657355 654a5c73 \bin;C:\Users\Jef
[7ffffa90] 65726666 75482079 5c676e61 44707041 ffreY Huang\AppData
[7ffffaa0] 5c617461 6d616f52 5c676e69 006d706e ata\Roaming\Npm.
[7ffffab0] 573d534f 6f646e69 4e5f7377 554e0054 OS=Windows_NT.NU
[7ffffac0] 5245424d 5f464f5f 434f5250 4f535345 MBER_OF_PROCESSES
[7ffffad0] 343d5352 474f4c00 45534e4f 52455652 RS=4.LOGONSERVERT
[7ffffae0] 4a5c5c3d 2d464645 4c004350 4c41434f =\JEFF-PC.LOCAL
[7ffffaf0] 44505041 3d415441 555c3a43 73726573 APPDATA=C:\Users
[7ffffb00] 66654a5c 79657266 61754820 415c676e \Jeffrey Huang\A
[7ffffb10] 61447070 4c5c6174 6c61636f 4d4f4800 ppData\Local.HOM
[7ffffb20] 54415045 555c3d48 73726573 66654a5c EPATH=\Users\Jef
[7ffffb30] 79657266 61754820 4800676e 44454d4f freY Huang.HOMED
[7ffffb40] 45564952 003a433d 454d4f48 5c3a433d RIVE=C:.HOME=C:\
[7ffffb50] 72657355 654a5c73 65726666 75482079 Users\Jeffrey Hu
[7ffffb60] 5c676e61 44707041 5c617461 6d616f52 ang\AppData\Roam
[7ffffb70] 5c676e69 5f425053 362e3631 5f504600 ing\SPB_16.6.FP_

[7ffffe20]	485f4f4e	5f54534f	43454843	4f4e3d4b	NO_HOST_CHECK=NO
[7ffffe30]	53504600	4f52425f	52455357	4553555f	.FPS_BROWSER_USE
[7ffffe40]	52505f52	4c49464f	54535f45	474e4952	R_PROFILE_STRING
[7ffffe50]	6665443d	746c7561	53504600	4f52425f	=Default.FPS_BRO
[7ffffe60]	52455357	5050415f	4f52505f	454c4946	WSER_APP_PROFILE
[7ffffe70]	5254535f	3d474e49	65746e49	74656e72	-STRING=Internet
[7ffffe80]	70784520	65726f6c	4f430072	5045434e	-Explorer.CONCEP
[7ffffe90]	4e495f54	445f5453	433d5249	724f5c3a	T_INST_DIR=C:\Or
[7ffffea0]	5c444143	4143724f	36315f44	4c5f362e	CAD\OrCAD_16.6_L
[7ffffeb0]	00657469	536d6f43	3d636570	575c3a43	ite.ComSpec=C:\W
[7ffffec0]	4f444e49	735c5357	65747379	5c32336d	INDOWS\system32\
[7ffffed0]	2e646d63	00657865	504d4f43	52455455	cmd.exe.COMPUTER
[7ffffee0]	454d414e	46454a3d	43502d46	6d6f4300	NAME=JEFF-PC.Com
[7ffffef0]	506e6f6d	72676f72	36576d61	3d323334	monProgramW6432=
[7fffff00]	505c3a43	72676f72	46206d61	73656c69	C:\Program Files
[7fffff10]	6d6f435c	206e6f6d	656c6946	6f430073	\Common Files.Com
[7fffff20]	6e6f6d6d	676f7250	466d6172	73656c69	monProgramFiles
[7fffff30]	36387828	3a433d29	6f72505c	6d617267	(x86)=C:\Program
[7fffff40]	6c694620	28207365	29363878	6d6f435c	Files (x86)\Com
[7fffff50]	206e6f6d	656c6946	6f430073	6e6f6d6d	mon Files.Common
[7fffff60]	676f7250	466d6172	73656c69	5c3a433d	ProgramFiles=C:\
[7fffff70]	676f7250	206d6172	656c6946	78282073	Program Files (x
[7fffff80]	5e293638	6d6d6f43	46206e6f	73656c69	86)\Common Files
[7fffff90]	53444300	544f4f52	5c3a433d	4143724f	.CDSROOT=C:\OrCA
[7fffffa0]	724f5c44	5f444143	362e3631	74694c5f	D\OrCAD_16.6_Lit
[7fffffb0]	50410065	54414450	3a433d41	6573555c	e.APPDATA=C:\Use
[7fffffc0]	4a5c7372	72666665	48207965	676e6175	rs\Jeffrey Huang
[7fffffd0]	7070415c	61746144	616f525c	676e696d	\AppData\Roaming
[7fffffe0]	4c4c4100	52455355	4f525053	454c4946	.ALLUSERSPROFILE
[7ffffff0]	5c3a433d	676f7250	446d6172	00617461	=C:\ProgramData.

Kernel data segment [90000000]..[90010000]

[90000000]	78452020	74706563	206e6f69	636f2000	Exception . oc
[90000010]	72727563	61206465	6920646e	726f6e67	curred and ignor
[90000020]	000a6465	495b2020	7265746e	74707572	ed.. [Interrupt
[90000030]	2000205d	4c545b20	20005d42	4c545b20] . [TLB]. [TL
[90000040]	20005d42	4c545b20	20005d42	64415b20	B]. [TLB]. [Ad
[90000050]	73657264	72652073	20726f72	69206e69	dress error in i
[90000060]	2f74736e	61746164	74656620	205d6863	nst/data fetch]
[90000070]	5b202000	72646441	20737365	6f727265	. [Address erro
[90000080]	6e692072	6f747320	205d6572	5b202000	r in store]. [
[90000090]	20646142	74736e69	74637572	206e6f69	Bad instruction
[900000a0]	72646461	5d737365	20200020	6461425b	address]. [Bad
[900000b0]	74616420	64612061	73657264	00205d73	data address].
[900000c0]	455b2020	726f7272	206e6920	63737973	[Error in sysc
[900000d0]	5d6c6c61	20200020	6572425b	6f706b61	all]. [Breakpo
[900000e0]	5d746e69	20200020	7365525b	65767265	int]. [Reserve
[900000f0]	6e692064	75727473	6f697463	00205d6e	d instruction].
[90000100]	5b202000	74697241	74656d68	6f206369	. [Arithmetic o
[90000110]	66726576	5d776f6c	20200020	6172545b	verflow]. [Tra
[90000120]	00205d70	5b202000	616f6c46	676e6974	p].. [Floating
[90000130]	696f7020	205d746e	20000000	6f435b20	point]... [Co
[90000140]	636f7270	005d3220	20000000	444d5b20	proc 2]... [MD
[90000150]	005d584d	575b2020	68637461	2020005d	MX]. [Watch].
[90000160]	63614d5b	656e6968	65686320	005d6b63	[Machine check].
[90000170]	00000000	5b202000	68636143	00005d65	... [Cache]..
[90000180]	90000024	90000033	9000003b	90000043	\$.3...;...C...
[90000190]	9000004b	90000071	9000008d	900000aa	K...q.....
[900001a0]	900000c0	900000d6	900000e6	90000100
[900001b0]	90000101	9000011a	90000124	90000125\$.%...
[900001c0]	90000139	9000013a	9000013b	90000148	9...:...;...H...
[900001d0]	90000149	9000014a	9000014b	90000154	I...J...K...T...
[900001e0]	9000015e	90000170	90000171	90000172	^...p...q...r...
[900001f0]	90000173	90000174	90000175	9000017f	s...t...u.....
[90000200]..[9000ffff]	00000000				

- The reason why index j is (size – 1) but not size is because the index number for the first element starts at zero instead of one. Hence, the total size of the array would go from zero to seven resulting in eight elements.

Code used for Assignment 4

```
# Jeffrey Huang

# Used in assignment 4

# Registers used: $t0 - used to hold the first number.
# $t1 - used to hold the second number.
# $t2 - used to hold the sum of the $t1 and $t2.
# $v0 - syscall parameter and return value.
# $a0 - syscall parameter.

.text

main:
## Get first number from user, put into $t0
    li $v0, 5          # load syscall read_int into $v0
    syscall            # make the syscall.
    move $t0, $v0      # move the number read into $t0

## Get second number from user, put into $t1.
    li $v0, 5          # load syscall read_int into $v0
    syscall            # make the syscall.
    move $t1, $v0      # move the number read into $t1
    add $t2, $t0, $t1  # compute the sum.

## Print out $t2.
    move $a0, $t2      # move the number to print into $a0.
    li $v0, 1          # load syscall print_int into $v0.
    syscall            # make the syscall.
    li $v0, 10         # syscall code 10 is for exit.
    syscall            # make the syscall.

# The importance and value of register $v0 in using syscall is to make sure that the system
# has a default and fixed value that will cause the program to recognize and end the program.
```

Assignment 4:

PC	=	400050	PC	=	400050	PC	=	400050
EPC	=	0	EPC	=	0	EPC	=	0
Cause	=	0	Cause	=	0	Cause	=	0
BadVAddr	=	0	BadVAddr	=	0	BadVAddr	=	0
Status	=	3000ff10	Status	=	3000ff10	Status	=	3000ff10
HI	=	0	HI	=	0	HI	=	0
LO	=	0	LO	=	0	LO	=	0
R0 [r0]	=	0	R0 [r0]	=	0	R0 [r0]	=	0
R1 [at]	=	0	R1 [at]	=	0	R1 [at]	=	0
R2 [v0]	=	a	R2 [v0]	=	a	R2 [v0]	=	a
R3 [v1]	=	0	R3 [v1]	=	0	R3 [v1]	=	0
R4 [a0]	=	3	R4 [a0]	=	4	R4 [a0]	=	c8
R5 [a1]	=	7ffff430	R5 [a1]	=	7ffff430	R5 [a1]	=	7ffff430
R6 [a2]	=	7ffff444	R6 [a2]	=	7ffff444	R6 [a2]	=	7ffff444
R7 [a3]	=	0	R7 [a3]	=	0	R7 [a3]	=	0
R8 [t0]	=	1	R8 [t0]	=	2	R8 [t0]	=	64
R9 [t1]	=	2	R9 [t1]	=	2	R9 [t1]	=	64
R10 [t2]	=	3	R10 [t2]	=	4	R10 [t2]	=	c8
R11 [t3]	=	0	R11 [t3]	=	0	R11 [t3]	=	0
R12 [t4]	=	0	R12 [t4]	=	0	R12 [t4]	=	0
R13 [t5]	=	0	R13 [t5]	=	0	R13 [t5]	=	0
R14 [t6]	=	0	R14 [t6]	=	0	R14 [t6]	=	0
R15 [t7]	=	0	R15 [t7]	=	0	R15 [t7]	=	0
R16 [s0]	=	0	R16 [s0]	=	0	R16 [s0]	=	0
R17 [s1]	=	0	R17 [s1]	=	0	R17 [s1]	=	0
R18 [s2]	=	0	R18 [s2]	=	0	R18 [s2]	=	0
R19 [s3]	=	0	R19 [s3]	=	0	R19 [s3]	=	0
R20 [s4]	=	0	R20 [s4]	=	0	R20 [s4]	=	0
R21 [s5]	=	0	R21 [s5]	=	0	R21 [s5]	=	0
R22 [s6]	=	0	R22 [s6]	=	0	R22 [s6]	=	0
R23 [s7]	=	0	R23 [s7]	=	0	R23 [s7]	=	0
R24 [t8]	=	0	R24 [t8]	=	0	R24 [t8]	=	0
R25 [t9]	=	0	R25 [t9]	=	0	R25 [t9]	=	0
R26 [k0]	=	0	R26 [k0]	=	0	R26 [k0]	=	0
R27 [k1]	=	0	R27 [k1]	=	0	R27 [k1]	=	0
R28 [gp]	=	10008000	R28 [gp]	=	10008000	R28 [gp]	=	10008000
R29 [sp]	=	7ffff42c	R29 [sp]	=	7ffff42c	R29 [sp]	=	7ffff42c
R30 [s8]	=	0	R30 [s8]	=	0	R30 [s8]	=	0
R31 [ra]	=	400018	R31 [ra]	=	400018	R31 [ra]	=	400018

1.

$$1 + 2 = 3$$

$$2 + 2 = 4$$

$$100 + 100 = 200$$

2. The value of register \$v0 in using syscall dictates what kind of action the system will take. Each value results in a different reading or printing method. If the incorrect number is loaded into the register, syscall will perform an action that the programmer does not desire.

3. Code for Question 3 is shown below.


```
# Jeffrey Huang
# Assignment 4 Question 3

.data
    buffer: .space 100
    ask: .asciiz "Please type a string:\n"
    tell: .asciiz "The string that you entered:\n"

.text

main:
    la $a0, ask      # Load and print string asking for string
    li $v0, 4        # load syscall print_string into $v0
    syscall          # make the syscall.
    li $v0, 8        # take user input
    la $a0, buffer   # load byte space into address
    li $a1, 100      # allot the byte space for string
    move $t0, $a0    # save string to t0
    syscall          # make the syscall.
    la $a0, tell     # print string with tell header
    li $v0, 4        # load syscall print_string into $v0
    syscall          # make the syscall.
    la $a0, buffer   # reload byte space to primary address
    move $a0, $t0    # primary address = t0 address (load pointer)
    li $v0, 4        # print string
    syscall          # make the syscall.
    li $v0, 10       # syscall code 10 is for exit.
    syscall          # make the syscall.
```

Output:

 Console

```
Please type a string:
Hello , world
The string that you entered:
Hello , world
|
```

Code used for Assignment 5

```
# Jeffrey Huang
# Assignment 5

.data
    buffer: .space 100
    ask_index: .asciiz "Please enter the number of integers you would like to multiply:\n"
    ask_number: .asciiz "Please enter the number that you would like to multiply:\n"
    tell: .asciiz "The product that you made:\n"

.text

main:
    li $t3, 1
## Getting the number of integers that the user wants to enter
    la $a0, ask_index      # Load and print string asking for string
    li $v0, 4              # load syscall print_string into $v0
    syscall                # make the syscall.
## Get first number from user, put into $t0 (number of integers)
    li $v0, 5              # load syscall read_int into $v0
    syscall                # make the syscall.
    move $t0, $v0          # move the number read into $t0
    #addi $t0, $t0, 1      # making number input the nuber of inputs.
    li $t1, 0              # initializing i = 0
    j loop                 # jump to loop

loop:
    bge $t1, $t0, END      # from i = 0 to n - 1 (n times)
    addi $t1, $t1, 1       # incrementing i by 1
    la $a0, ask_number     # Load and print string asking for string
    li $v0, 4              # load syscall print_string into $v0
    syscall                # make the syscall.
    li $v0, 5              # load syscall read_int into $v0
    syscall                # make the syscall.
    move $t2, $v0          # move the number read into $t0
    mul $t3, $t3, $t2      # $t3 = $t3 * $t2
    j loop                 # jump to loop

END:
    la $a0, tell           # print string with tell header
    li $v0, 4              # load syscall print_string into $v0
    syscall                # make the syscall.
    move $a0, $t3          # move the number to print into $a0.
    li $v0, 1              # load syscall print_int into $v0.
    syscall                # make the syscall
    li $v0, 10             # syscall code 10 is for exit.
    syscall                # make the syscall.
```

Output:

```
Please enter the number of integers you would like to multiply:
5
Please enter the number that you would like to multiply:
2
Please enter the number that you would like to multiply:
2
Please enter the number that you would like to multiply:
2
Please enter the number that you would like to multiply:
2
Please enter the number that you would like to multiply:
2
The product that you made:
32
```

Code used for Assignment 6

```
# Jeffrey Huang
# Assignment 6

.data
ask_index: .asciiz "Please enter the value of n that you would like to get use: \n"
tell_result: .asciiz "Result of a[n] = "
.text

main:
    la $a0, ask_index      # Load and print string asking for string
    li $v0, 4              # load syscall print_string into $v0
    syscall                # make the syscall.
    li $v0, 5              # load syscall read_into into $v0
    syscall                # make the syscall.
    move $t0, $v0          # move the number read into $t0
    addi $t0, $t0, 1       # making number input the nuber of inputs.
    li $t1, 3              # initializing i = 2
    li $t2, 0              # a[0] = 0
    li $t3, 1              # a[1] = 1
    li $t4, 1              # a[2] = 1
    j loop

loop:
    bge $t1, $t0, END      # for loop from 3 to n
    addi $t1, $t1, 1       # increment the counter
    add $t5, $t2, $t3      # adding a[n-3] and a[n-2]
    add $t5, $t5, $t4      # adding $t5 to a[n-1]
    move $t2, $t3          # moving the $t3 into $t2
    move $t3, $t4          # moving the $t4 into $t3
    move $t4, $t5          # moving the $t5 into $t4
    j loop                 # jump to loop

END:
    la $a0, tell_result    # print string with tell header
    li $v0, 4              # load syscall print_string into $v0
    syscall                # make the syscall.
    move $a0, $t4          # move the number to print into $a0.
    li $v0, 1              # load syscall print_int into $v0.
    syscall                # make the syscall
    li $v0, 10             # syscall code 10 is for exit.
    syscall                # make the syscall.
```

Output:

```
Please enter the value of n that you would like to get use:
9
Result of a[n] = 81
```

Registers:

		R7	[a3] = 0	
PC	= 400098	R8	[t0] = a	
EPC	= 0	R9	[t1] = a	
Cause	= 0	R10	[t2] = 18	
BadVAddr	= 0	R11	[t3] = 2c	R22 [s6] = 0
Status	= 3000ff10	R12	[t4] = 51	R23 [s7] = 0
		R13	[t5] = 51	R24 [t8] = 0
HI	= 0	R14	[t6] = 0	R25 [t9] = 0
LO	= 0	R15	[t7] = 0	R26 [k0] = 0
		R16	[s0] = 0	R27 [k1] = 0
R0	[r0] = 0	R17	[s1] = 0	R28 [gp] = 10008000
R1	[at] = 10010000	R18	[s2] = 0	R29 [sp] = 7ffff420
R2	[v0] = a	R19	[s3] = 0	R30 [s8] = 0
R3	[v1] = 0	R20	[s4] = 0	R31 [ra] = 400018
R4	[a0] = 51	R21	[s5] = 0	
R5	[a1] = 7ffff424			
R6	[a2] = 7ffff43c			

Assignment 7:

```
add $t1, $s4($s1), $s4($s2)
mul $t1, $t1, $s4($s2)
sw $t1, $s5($s2)
```

Assignment 8:

```
# Jeffrey Huang
# Used for assignment 8

#ex5.asm

.data 0x10000000
ask: .asciiz "\nEnter a number:"
ans: .asciiz "Answer: "
.text 0x00400000
.globl main

main:
    li $v0, 4
    la $a0, ask           # Loads the ask string
    syscall              # Display the ask string
    li $v0, 5             # Read the input
    syscall
    move $t0, $v0         # n = $v0, Move the user input
    addi $t1, $0, 0       # i = 0
    addi $t2, $0, 189     # ans = 189, Starting case (m = 0)
    li $t3, 2

loop:
    beq $t1, $t0, END     # from i = 0 to n-1 (n times)
    addi $t1, $t1, 1      # i = i + 1
    div $t2, $t3          # LO = ans / 2, The result is stored on LO
    mflo $t2              # ans = LO, Copies the content of LO to t2
    j loop

END:
    li $v0, 4
    la $a0, ans           # Loads the ans string
    syscall
    move $a0, $t2         # Loads the answer
    li $v0, 1
    syscall
    li $v0, 10
    syscall

# Any value other than 0 will not result in 189 being divided by 2 over
# and over the number of times greater than 0
# ie: input 1 -> 94
# ie: input 2 -> 47
# ie: input 3 -> 23
# ie: input 4 -> 11
# Once the input value is greater than or equal to 8, the value returned
# will be equivalent to 0 regardless of how many times it needs to be
# divided again.
# Hence the solution to this program is 0.
```

Output:

```
Enter a number:0
Answer: 189
Enter a number:1
Answer: 94
Enter a number:2
Answer: 47
Enter a number:3
Answer: 23
Enter a number:4
Answer: 11
```