

CPE 325: Intro to Embedded Computer System

Lab11

REVERSE ENGINEERING.

Submitted by: Dan Otieno

Date of Experiment: 04/04/22

Report Deadline: 04/11/22

Demonstration Deadline: 04/12/22

Introduction

This report introduces us to the concept of software reverse engineering. The purpose of this lab is to learn how to use some of the functionalities of the MSP430FG418 board to analyze software and determine its components. The tests performed in this lab will be to crack a binary file and disassemble a hexadecimal text file so we can extract assembly code.

Theory

Topic 1: ELF.

ELF stands for Executable and Linkable Format, used for executable files, object files, shared libraries and core dumps. It is not bound by the ISA in the MSP430 or operating systems. The ELF contains the following segments:

- A file header (ELF file header).
- Program header table which tells the loader how to create a process image in memory.
- Section header table which describes sections that contain data referred to by entries in the program and section header tables.
- Segments that contain details needed for the execution of runtime.
- Sections that contain linking and relocation information.

They are also divided into sections that have a Name and Type, requested memory location at runtime and permissions (read, write etc).

Topic 2: NAKEN UTILITY.

Naken Utility is a disassembler developed by Michael Kohn and Joe Davisson that disassembles a hex file (passed as an input) and produces assembly code that can be analyzed, and reverse engineered. We use this tool on this lab to reverse engineer a provided hex file.

Topic 3: MSP430 FLASHER.

This is a utility program used to flash a hex file with executable on a target platform. The flasher retrieves code from the platform and stores it into an output file in hex or text format. Once the output is extracted, we can use the naked utility, discussed above, to strip it and extract the assembly file.

Results & Observation

Program 1:

Program Description:

The first part of this lab was to save a downloadable .out file that was provided and use the methods learned in the lab to determine the correct password from a list of possible passwords in the file. The steps followed to achieve that are documented below:

- STEP 1: File downloaded from Canvas and saved in local directory (used downloads folder).
- STEP 2: On CCS, I created a new project, connected the MSP430 board, and loaded the .out file to debug.
- STEP 3: MobaXterm launched, serial connection made with a baud rate of 115,200. Noted the display on the screen prompting for a password.
- STEP 4: Launched a command prompt window and typed in the following commands: `msp430-elf-readelf -a Lab11_crack_me_danotieno.out` to display the ELF Header, section headers, file attributes, passwords.

C:\Windows\System32\cmd.exe

Microsoft Windows [Version 10.0.22000.376]

(c) Microsoft Corporation. All rights reserved.

C:\Users\dpo0002\Downloads>msp430-elf-readelf -a Lab11_crack_me_danotieno.out

ELF Header:

Magic: 7f 45 4c 46 01 01 01 ff 00 00 00 00 00 00 00
Class: ELF32
Data: 2's complement, little endian
Version: 1 (current)
OS/ABI: Standalone App
ABI Version: 0
Type: EXEC (Executable file)
Machine: Texas Instruments msp430 microcontroller
Version: 0x1
Entry point address: 0x349c
Start of program headers: 52 (bytes into file)
Start of section headers: 2260 (bytes into file)
Flags: 0xb: architecture variant: MSP430x11
Size of this header: 52 (bytes)
Size of program headers: 32 (bytes)
Number of program headers: 5
Size of section headers: 40 (bytes)
Number of section headers: 57
Section header string table index: 56

Section Headers:

[Nr]	Name	Type	Addr	Off	Size	ES	Flg	Lk	Inf	Al
[0]		NULL	00000000	000000	000000	00		0	0	0
[1]	.bss	NOBITS	00000000	000000	000000	00	WA	0	0	1
[2]	.data	NOBITS	00001100	0000d4	000001	00	WA	0	0	1
[3]	.TI.noinit	NOBITS	00000000	000000	000000	00	p	0	0	1
[4]	.sysmem	NOBITS	00000000	000000	000000	00		0	0	1
[5]	.stack	NOBITS	000030b0	0000d4	000050	00	WA	0	0	4
[6]	.text	PROGBITS	00003100	000124	000438	00	AX	0	0	2
[7]	.text:_isr	PROGBITS	000035f2	000616	000008	00	AX	0	0	2
[8]	.cinit	PROGBITS	000035fa	00061e	00000e	00	A	0	0	2
[9]	.const	PROGBITS	00003538	00055c	0000ba	00	A	0	0	2
[10]	.bss:signature	NOBITS	00000000	000000	000000	00		0	0	1
[11]	.cio	NOBITS	00000000	000000	000000	00		0	0	1
[12]	.pinit	NOBITS	00000000	000000	000000	00		0	0	1
[13]	.binit	PROGBITS	00003100	000124	000000	00	A	0	0	2
[14]	.init_array	INIT_ARRAY	00000000	000652	000000	04	WA	0	0	1
[15]	.mspabi.exidx	NOBITS	00000000	000000	000000	08	A	0	0	1
[16]	.mspabi.exstab	NOBITS	00000000	000000	000000	00	A	0	0	1
[17]	.TI.ramfunc	NOBITS	00000000	000000	000000	00		0	0	1
[18]	.infoA	NOBITS	00000000	000000	000000	00		0	0	1
[19]	.infoB	NOBITS	00000000	000000	000000	00		0	0	1
[20]	.int00	NOBITS	00000000	000000	000000	00		0	0	1
[21]	.int01	NOBITS	00000000	000000	000000	00		0	0	1
[22]	.int02	NOBITS	00000000	000000	000000	00		0	0	1
[23]	.int03	NOBITS	00000000	000000	000000	00		0	0	1
[24]	.int04	NOBITS	00000000	000000	000000	00		0	0	1
[25]	.int05	NOBITS	00000000	000000	000000	00		0	0	1
[26]	.int06	NOBITS	00000000	000000	000000	00		0	0	1
[27]	.int07	NOBITS	00000000	000000	000000	00		0	0	1
[28]	.int08	NOBITS	00000000	000000	000000	00		0	0	1
[29]	.int09	NOBITS	00000000	000000	000000	00		0	0	1
[30]	.int10	NOBITS	00000000	000000	000000	00		0	0	1
[31]	.int11	NOBITS	00000000	000000	000000	00		0	0	1
[32]	.int12	NOBITS	00000000	000000	000000	00		0	0	1
[33]	.int13	NOBITS	00000000	000000	000000	00		0	0	1
[34]	DAC12	PROGBITS	0000ffdc	00062e	000002	00	A	0	0	1
[35]	DMA	PROGBITS	0000ffde	000630	000002	00	A	0	0	1

C:\Windows\System32\cmd.exe

C (compressed), x (unknown), o (OS specific), E (exclude),
p (processor specific)

There are no section groups in this file.

Program Headers:

Type	Offset	VirtAddr	PhysAddr	FileSiz	MemSiz	Flg	Align
LOAD	0x000000	0x00001100	0x00001100	0x00000	0x00001	RW	0x1
LOAD	0x0000d4	0x000030b0	0x000030b0	0x00050	0x00050	RW	0x4
LOAD	0x000124	0x00003100	0x00003100	0x00508	0x00508	R E	0x2
LOAD	0x00062c	0x0000ffbe	0x0000ffbe	0x00002	0x00002	R	0x1
LOAD	0x00062e	0x0000ffdc	0x0000ffdc	0x00024	0x00024	R	0x2

Section to Segment mapping:

Segment Sections...

00	.data
01	.stack
02	.text .text:_isr .cinit .const .binit
03	\$fill000
04	DAC12 DMA BASICTIMER PORT2 USART1TX USART1RX PORT1 TIMERA1 TIMERA0 ADC12 USCIAB0TX USCIAB0RX WDT COMPARATORA TIMERB1 TIMERB0 NMI .reset

There is no dynamic section in this file.

There are no relocations in this file.

The decoding of unwind sections for machine type Texas Instruments msp430 microcontroller is not currently supported.

No version information found in this file.

Attribute Section: TI

File Attributes

Unknown attribute:

```
0x00000000 054c696e 6b657200 08160a07 0c018602 .Linker.....
0x00000010 02870230 31323334 35363738 39303141 ...012345678901A
0x00000020 42434445 46474849 4a303131 31313131 BCDEFGHIJ0111111
0x00000030 31313131 30313130 30303030 30303030 11110110000000000
0x00000040 30313030 30303030 30303030 30303030 0100000000000000
0x00000050 30303030 30008802 01900201 92020194 00000.....
0x00000060 02019602 01 .....

```

Attribute Section: mspabi

File Attributes

Tag_ISA: MSP430
Tag_Code_Model: Small
Tag_Data_Model: Small
<unknown tag 10>: 3 (0x3)

C:\Users\dpo0002\Downloads>msp430-elf-strings -a Lab11_crack_me_danotieno.out

```
5>@!
5>@!
C>@0
A<P0
C>@0
A<P0
A<P0
$|0*
/Q10
LxJ C
~J>
$|JLL
#00$51
-0,A
6500$5=
/Q10

```

C:\Windows\System32\cmd.exe

```
$|@*  
/QlO  
LxJ      C  
~J>  
$|JLL  
#0@$51  
-O,A  
650@$5=  
/QlO  
#!R0A  
\0A1@  
LnOkMLN  
'0A!  
!S0A  
#0A4A5A6A7A8A9A:A0A  
C0A0A  
whatdoyousee  
circumambulate  
aDreamySabbath  
TheCityof  
afternoon  
Correct!  
Incorrect password!  
/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o/o  
What is your guessed password?  
CPE325_user  
Linker  
012345678901ABCDEFGHJIj0111111111101100000000001000000000000000000  
mspabi  
.shstrtab  
.bss  
.data  
.TI.noinit  
.system  
.stack  
.text  
.text:_isr  
.cinit  
.const  
.bslsignature  
.cio  
.pininit  
.bininit  
.init_array  
.mspabi.exidx  
.mspabi.extab  
.TI.ramfunc  
.infoA  
.infoB  
.int00  
.int01  
.int02  
.int03  
.int04  
.int05  
.int06  
.int07  
.int08  
.int09  
.int10  
.int11  
.int12  
.int13
```

-
- ```
C:\Users\System32\cmd.exe
$!$!>
/Q10
c
->>
$!JL
mgq$1
-Q,A
mgng$5=
/Q10
#ROA
AA1g
-ORLN
'OKI
isba
HHAASAG6A7ABA9A:ABA
CA8AA
whatdoyousee
circumulate
ADreamySabbath
Heckifyor
afternoon
Correct!
Incorrect password!
=====
What is your guessed password?
C@E7P_usen
Linker
012345678901ABCDEF GHI7011111111101100000000001000000000000000000000000
Linker
shstrtab
.bss
.data
$.TI.noinit
.system
.stack
.text
$.text_isr
.cinit
.const
.mdisignature
.cio
.pinit
.sinit
.init_array
.mspabi.exidx
.mspabi.extab
$.TI.ramfunc
.infoA
.infoB
.int00
.int01
.int02
.int03
.int04
.int05
.int06
.int07
.int08
.int09
.int10
.int11
.int12
.int13
```

- Magic Number: 7f 45 4c 46 01 01 ff 00 00 00 00 00 00 00.
- Class of .out file: ELF32.
- Machine the file was built for: Texas Instruments msp430 microcontroller.
- Size of the header: 52 bytes.
- Number of section headers: 57. (Verified by running the program twice).

a. Having programmed the Hex file using the MSP430, this was the output in the command window, I forgot to copy the HEX txt file in my USB thumb drive to include in this report, this can demonstrated in the lab.

```

C:\Users\dpo0002\Downloads>MSP430Flasher.exe -n MSP43FG4618 -w Lab11_reverse_me.txt -v -z [Vcc]
* -----/|----- *
* /| *
* /-| MSP Flasher v1.3.20 *
* | / *
* -|/----- *
* /|----- *
*
* Evaluating triggers...done
* Checking for available FET debuggers:
* Found USB FET @ COM17 <- Selected
* Initializing interface @ COM17...done
* Checking firmware compatibility:
* FET firmware is up to date.
* Reading FW version...done
* Setting VCC to 3000 mV...done
* Accessing device...done
* Reading device information...
* Warning: Found device does not match -n selection:
- Selected: MSP43FG4618
- Found: MSP430FG4618
- Continue? (Y/N): y
* Loading file into device...done
* Verifying memory (Lab11_reverse_me.txt)...done
*
* -----
* Arguments : -n MSP43FG4618 -w Lab11_reverse_me.txt -v -z [Vcc]
* -----
* Driver : loaded
* Dll Version : 31400000
* FwVersion : 31100001
* Interface : TIUSB
* HwVersion : U 1.40
* JTAG Mode : AUTO
* Device : MSP430FG4618
* EEM : Level 3, ClockCntrl 2
* Erase Mode : ERASE_ALL
* Prog.File : Lab11_reverse_me.txt
* Verified : TRUE
* BSL Unlock : FALSE
* InfoA Access: FALSE
* VCC ON : 3000 mV
* -----
* Starting target code execution...done
* Disconnecting from device...done
*
* -----
* Driver : closed (No error)
* -----
*/

```

In addition to the HEX file, I also tested the LEDs on the MSP430 board. S1 press toggled the yellow LED on and S2 press toggled the green LED on. However, these presses only worked once, meaning, the outputs were alternating, if, for instance a switch was pressed to turn on the LED, the second LED had to be turned on before we could get the first one to turn on again.

- b. From the observation board, I can guess that the program toggles the LEDs in the MSP430 board.
- c. Once the step above was completed, the following next two steps were taken to run the naked utility:



```

C:\Users\dpo0002\Downloads>MSP430Flasher.exe -r [Lab11_reverse_me.txt, MAIN]
* -----/|-----*
* /|_ *
* /_/_/ MSP Flasher v1.3.20 *
* |/_/ *
* -----/|-----*
*
* Evaluating triggers...done
* Checking for available FET debuggers:
* Found USB FET @ COM17 <- Selected
* Initializing interface @ COM17...done
* Checking firmware compatibility:
* FET firmware is up to date.
* Reading FW version...done
* Setting VCC to 3000 mV...done
* Accessing device...done
* Reading device information...done
* Dumping memory from MAIN into Lab11_reverse_me.txt...done
*
* -----
* Arguments : -r [Lab11_reverse_me.txt, MAIN]
* -----
* Driver : loaded
* Dll Version : 31400000
* FwVersion : 31100001
* Interface : TIUSB
* HwVersion : U 1.40
* JTAG Mode : AUTO
* Device : MSP430FG4618
* EEM : Level 3, ClockCntrl 2
* Read File : Lab11_reverse_me.txt (memory segment = MAIN)
* VCC OFF
* -----
* Powering down...done
* Disconnecting from device...done
*
* -----
* Driver : closed (No error)
* -----
*/

```

## Program Flowchart:

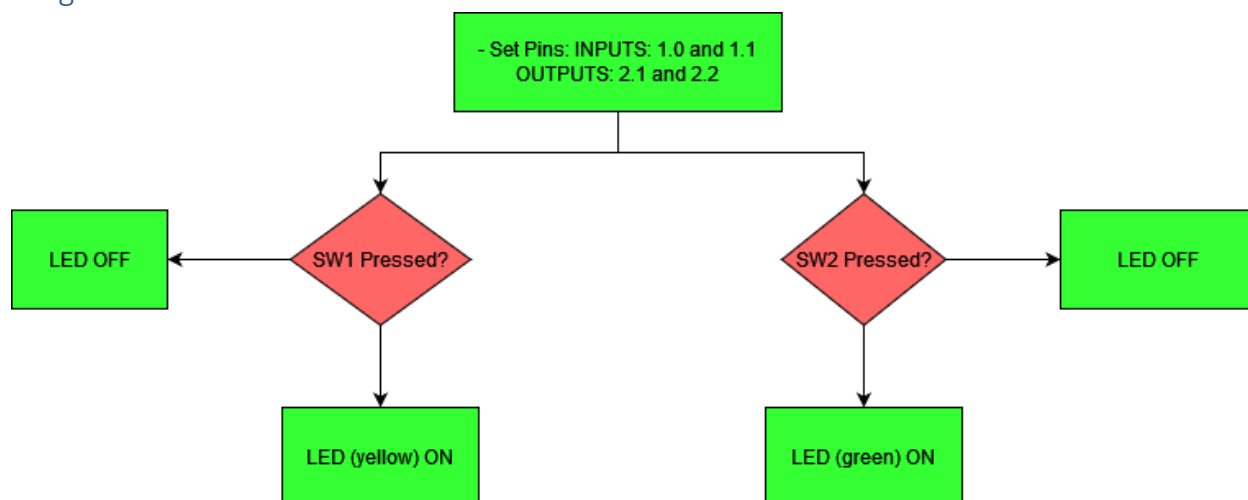


Figure 01: Program flowchart.

## Appendix

Table 01: Commented Assembly output.

| naken_util - by Michael Kohn<br>Joe Davisson<br>Web: <a href="http://www.mikekohn.net/">http://www.mikekohn.net/</a><br>Email: <a href="mailto:mike@mikekohn.net">mike@mikekohn.net</a> |        |                        |                      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|------------------------|----------------------|
| Version:                                                                                                                                                                                |        |                        |                      |
| Loaded ti_txt Lab11_reverse_me.txt from 0x3100 to 0x1ffff                                                                                                                               |        |                        |                      |
| Type help for a list of commands.                                                                                                                                                       |        |                        |                      |
| Addr                                                                                                                                                                                    | Opcode | Instruction            | Cycles               |
| 0x3100:                                                                                                                                                                                 | 0x40b2 | mov.w #0x5a80, &0x0120 | 5; STOP Watchdog     |
| Timer.                                                                                                                                                                                  |        |                        |                      |
| 0x3102:                                                                                                                                                                                 | 0x5a80 |                        |                      |
| 0x3104:                                                                                                                                                                                 | 0x0120 |                        |                      |
| 0x3106:                                                                                                                                                                                 | 0xd2e2 | bis.b #4, &0x002a      | 4; Bit set.          |
| 0x3108:                                                                                                                                                                                 | 0x002a |                        |                      |
| 0x310a:                                                                                                                                                                                 | 0xd3e2 | bis.b #2, &0x002a      | 4; Bit set.          |
| 0x310c:                                                                                                                                                                                 | 0x002a |                        |                      |
| 0x310e:                                                                                                                                                                                 | 0xc2e2 | bic.b #4, &0x0029      | 4; Bit clear (P2.2). |
| 0x3110:                                                                                                                                                                                 | 0x0029 |                        |                      |
| 0x3112:                                                                                                                                                                                 | 0xc3e2 | bic.b #2, &0x0029      | 4; Bit clear (P2.1). |
| 0x3114:                                                                                                                                                                                 | 0x0029 |                        |                      |
| 0x3116:                                                                                                                                                                                 | 0xc3d2 | bic.b #1, &0x0022      | 4; Bit clear (P1.0). |
| 0x3118:                                                                                                                                                                                 | 0x0022 |                        |                      |
| 0x311a:                                                                                                                                                                                 | 0xc3e2 | bic.b #2, &0x0022      | 4; Bit clear (P1.1). |
| 0x311c:                                                                                                                                                                                 | 0x0022 |                        |                      |
| 0x311e:                                                                                                                                                                                 | 0xb3d2 | bit.b #1, &0x0020      | 4; Check if SW1      |
| pressed.                                                                                                                                                                                |        |                        |                      |
| 0x3120:                                                                                                                                                                                 | 0x0020 |                        |                      |

|                                                        |                     |
|--------------------------------------------------------|---------------------|
| 0x3122: 0x23fd jne 0x311e (offset: -6)                 | 2; If not, jmp back |
| to instruction at 0x311e to check for SW2 via polling. |                     |
| 0x3124: 0x120d push.w r13                              | 3; Push R13 to      |
| stack.                                                 |                     |
| 0x3126: 0x403d mov.w #0x1a08, r13                      | 2; Mov #0x1a08 into |
| R13.                                                   |                     |
| 0x3128: 0x1a08                                         |                     |
| 0x312a: 0x831d sub.w #1, r13                           | 1; Subtract 1 from  |
| R13.                                                   |                     |
| 0x312c: 0x23fe jne 0x312a (offset: -4)                 | 2; JNE to 0x314c,   |
| and keep decrementing by 1.                            |                     |
| 0x312e: 0x413d pop.w r13 -- mov.w @SP+, r13            | 2; Pop R13 from     |
| Stack.                                                 |                     |
| 0x3130: 0x4303 nop -- mov.w #0, CG                     | 1; Delay cycle.     |
| 0x3132: 0xd3e2 bis.b #2, &0x0029                       | 4;                  |
| 0x3134: 0x0029                                         |                     |
| 0x3136: 0xb3d2 bit.b #1, &0x0020                       | 4; Check if P2IN is |
| pressed.                                               |                     |
| 0x3138: 0x0020                                         |                     |
| 0x313a: 0x27fd jeq 0x3136 (offset: -6)                 | 2; JNE to 0x3158,   |
| sw2 polling.                                           |                     |
| 0x313c: 0xc3e2 bic.b #2, &0x0029                       | 4; Clear bit P2.1.  |
| 0x313e: 0x0029                                         |                     |
| 0x3140: 0xb3e2 bit.b #2, &0x0020                       | 4; Check if P1IN    |
| pressed.                                               |                     |
| 0x3142: 0x0020                                         |                     |
| 0x3144: 0x23fd jne 0x3140 (offset: -6)                 | 2; JNE to 0x3140,   |
| check for SW2 press.                                   |                     |
| 0x3146: 0x120d push.w r13                              | 3; Push R13 onto    |
| stack.                                                 |                     |
| 0x3148: 0x403d mov.w #0x1a08, r13                      | 2; Mov #0x1a08 into |
| R13.                                                   |                     |
| 0x314a: 0x1a08                                         |                     |
| 0x314c: 0x831d sub.w #1, r13                           | 1; subtract 1 from  |
| R13.                                                   |                     |
| 0x314e: 0x23fe jne 0x314c (offset: -4)                 | 2; JNE to 0x314c,   |
| decrement by 1.                                        |                     |
| 0x3150: 0x413d pop.w r13 -- mov.w @SP+, r13            | 2                   |
| 0x3152: 0x4303 nop -- mov.w #0, CG                     | 1; delay cycle.     |
| 0x3154: 0xd2e2 bis.b #4, &0x0029                       | 4; Turn on LED at   |
| P2.2.                                                  |                     |
| 0x3156: 0x0029                                         |                     |
| 0x3158: 0xb3e2 bit.b #2, &0x0020                       | 4; Turn on LED at   |
| P2.1.                                                  |                     |
| 0x315a: 0x0020                                         |                     |
| 0x315c: 0x27fd jeq 0x3158 (offset: -6)                 | 2; JNE to 0x3158,   |
| polling for SW2 press.                                 |                     |
| 0x315e: 0xc2e2 bic.b #4, &0x0029                       | 4; Clear bit for    |
| P2DIR.                                                 |                     |
| 0x3160: 0x0029                                         |                     |
| 0x3162: 0x3fdd jmp 0x311e (offset: -70)                | 2; JMP to infinite  |
| loop.                                                  |                     |
| 0x3164: 0x4031 mov.w #0x3100, SP                       | 2; Mov 0x3100 onto  |
| stack pointer.                                         |                     |
| 0x3166: 0x3100                                         |                     |
| 0x3168: 0x12b0 call #0x317e                            | 5; Call subroutine. |
| 0x316a: 0x317e                                         |                     |

|                                        |                     |
|----------------------------------------|---------------------|
| 0x316c: 0x430c mov.w #0, r12           | 1; Mov 0 into R12.  |
| 0x316e: 0x12b0 call #0x3100            | 5; Call instruction |
| at 0x3100 (start of program).          |                     |
| 0x3170: 0x3100                         |                     |
| 0x3172: 0x431c mov.w #1, r12           | 1; Mov 1 into R12.  |
| 0x3174: 0x12b0 call #0x3178            | 5; Call end of      |
| program (0x449a).                      |                     |
| 0x3176: 0x3178                         |                     |
| 0x3178: 0x4303 nop -- mov.w #0, CG     | 1; Delay cycle.     |
| 0x317a: 0x3fff jmp 0x317a (offset: -2) | 2; Jump to infinite |
| loop.                                  |                     |
| 0x317c: 0x4303 nop -- mov.w #0, CG     | 1; Delay cycle.     |
| 0x317e: 0x431c mov.w #1, r12           | 1; Move 1 into R12. |
| 0x3180: 0x4130 ret -- mov.w @SP+, PC   | 3; Reti instruction |
| to exit subroutine.                    |                     |
| 0x3182: 0xd032 bis.w #0x0010, SR       | 2; Set bit 2 of SR. |
| 0x3184: 0x0010                         |                     |
| 0x3186: 0x3ffd jmp 0x3182 (offset: -6) | 2; Jump to infinite |
| loop.                                  |                     |
| 0x3188: 0x4303 nop -- mov.w #0, CG     | 1; delay cycle.     |
| 0x318a: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x318c: 0xffff                         |                     |
| 0x318e: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x3190: 0xffff                         |                     |
| 0x3192: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x3194: 0xffff                         |                     |
| 0x3196: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x3198: 0xffff                         |                     |
| 0x319a: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x319c: 0xffff                         |                     |
| 0x319e: 0xffff and.b @r15+, -1(r15)    | 5                   |
| 0x31a0: 0xffff                         |                     |
| 0x31a2: 0xffff and.b @r15+, -1(r15)    | 5                   |

*Your next code goes here, if any.*

Table ##: Program 2 source code

```

/*-----
 * File: Lab01_P2.c
 * Function: ...
 * Description: ...
 * Input: ...
 * Output: ...
 * Author(s): Your Name
 * Date: ...
 -----/
#include <msp430.h>
#include <stdio.h>

int main(){
 ...
}

```

**Make sure to submit a single pdf file only**