Homework 1 Report  
 (Reverse engineering the machine code for GPIO pin toggle program)

Submitted by: Kripa Varma  
Date : 4/12/2015

# Summary

Files committed to git : hw1.c, hw1.h, types.h, makefile, hw1.elf, hw1\_dmp.txt, Homework1\_Report.docx  
 Compiled and obtained object dump using : Windows based Linaro tool chain for arm

# The machine code to assembly translation of while loop

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Inst. #** | **addr** | **m/c code word** | **assembly** | **<cond><opcode> cond = 0xe (always)** | **<Rd>**  **Dest Reg** | **<Rn>**  **Src Reg** | **Other operands** |
| **Start of while loop :** | | | | | | | |
| 1 | 8288 | e51b2010 | ldr r2, [fp, #-16]  [[1]](#load_store) | <opcode>=0x51  Load r2 word pre-idx  subtract imm offset  r3<-addr[fp-16] | r2(0x2) | fp(0xb) | offset\_12 = 0x10 |
| 2 | 828c | e5d23000 | ldrb r3, [r2]  [[1]](#load_store) | <opcode>=0x5d  Load byte pre- idx add imm offset  r3<-byte at addr[r2] | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 3 | 8290 | e3833040 | orr r3, r3, #64  [[2]](#orr) | <opcode>=0x38  OR immediate  r3<-r3|0x40 | r3(0x3) | r3(0x3) | imm\_8= x40  rotate\_imm=0 |
| 4 | 8294 | e5c23000 | strb r3, [r2]  [[1]](#load_store) | <opcode>=0x5c  Store byte pre- idx add imm offset  r3->byte at addr[r2] | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 5 | 8298 | e51b2010 | ldr r2, [fp, #-16] | Same as #1 | r2(0x2) | fp(0xb) | offset\_12 = 0x10 |
| 6 | 829c | e5d23000 | ldrb r3, [r2] | Same as #2 | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 7 | 82a0 | e3c33040 | bic r3, r3, #64  [[3]](#bic) | <opcode>=0x3c  Bit Clear  r3<-r3&!0x40 | r3(0x3) | r3(0x3) | imm\_8= x40  rotate\_imm=0 |
| 8 | 82a4 | e5c23000 | strb r3, [r2] | Same as #4 | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 9 | 82a8 | eafffff6 | b 8288  [[4]](#branch) | Unconditional branch, Do not store RA | 2s complement offset operand: fffff6  Sign extending to 30 bits : 3ffffff6  Shifting left by 2 to align to word:ffffffd8  Add to PC, which is 82b0 now : 8288 | | |

# Generic syntax of instructions used in the program

**[1] LOAD/STORE Immediate offset Encoding**

LDR|STR{<cond>} Rd, [<Rn>, #+/-<offset\_12>]



P (0/1) : post/pre indexed

U (0/1) : Offset Sub/Add

B (0/1) : Word/Byte access

W(0/1): BaseReg unchanged/Addr written back to BaseReg

L (0/1): Load/Store

**[2] ORR Syntax: (Logical OR)**

Rd := Rn OR shifter\_operand



I (0/1) : Register/Immediate shifter operand

S (0/1) : CPSR Unchanged/ CPSR Update after execution

**[3] BIC Syntax: ( Bit Clear)**

Rd := Rn AND NOT(shifter\_operand)



**[4] B (Branch)**

B{L}{<cond>} <target\_address>



L (0/1) : Store / Do not Store return address

<target\_address>:

Specifies the address to branch to. The branch target address is calculated by:

1. Sign-extending the 24-bit signed (two's complement) immediate to 30 bits.

2. Shifting the result left two bits to form a 32-bit value.

3. Adding this to the contents of the PC, which contains the address of the branch

instruction plus 8 bytes.

# Hours:

4 hours : GPIO related reading, coding, makefile.  
5-6 hrs : Study instructions encoding and documenting the decoding of machine code.

# References:

1. UM10360 LPC176x/5x User manual ; Section 9.5 – GPIO Registers
2. ARM Architecture Reference Manual ; Section A4, A5