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

# Generic syntax of instructions used in the program

1. MOV/MVN/ADD/SUB/BIC/ORR Immediate Encoding:



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

1. LOAD/STORE Immediate offset Encoding



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

# The machine code to assembly translation

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Inst. #** | **addr** | **m/c code word** | **assembly** | **<cond><opcode> cond = 0xe (always)** | **<Rd>**  **Dest Reg** | **<Rn>**  **Src Reg** | **Other operands** |
| 1 | 821c | e52db004 | push {fp}  (str fp, [sp, #-4]!) | <opcode>=0x52  Store word pre-idx subtract imm offset  Write back BaseReg,  fp-> addr [sp-4] | store reg  = fp(0xb) | base reg  =sp(0xd) | offset\_12 = 0x4 |
| 2 | 8220 | e28db000 | add fp, sp, #0 | <opcode>=0x28  Add immediate  fp<-sp+0 | Dest reg= fp(0xb) | 1st oprnd = sp(0xd) | imm \_8 =0x0  rotate\_imm=0 |
| 3 | 8224 | e24dd014 | sub sp, sp, #20 | <opcode>=0x24  Subtract immediate  sp<-sp-20 | 0xb = sp | 0xd = sp | imm\_8= 0x14  rotate\_imm=0 |
| **Init pointers:** | | | | | | | |
| 4 | 8228 | e3a03000 | mov r3, #0 | <opcode>=0x3a  mov immediate  r3<-0 | 0x3 = r3 | - | 0x0 |
| 5 | 822c | e50b3008 | str r3, [fp, #-8] | <opcode>=0x50  Store word pre-idx  subtract imm offset,  r3->addr [fp-8] | store reg = r3(0x3) | base reg = fp(0xb) | offset\_12 = 0x8 |
| 6 | 8230 | e3a03000 | mov r3, #0[5] | Same as #4 | Dest reg r3(0x3) | - | 0x0 |
| 7 | 8234 | e50b300c | str r3, [fp, #-12] | Same as #5 |  |  | offset\_12 = 0xc |
| 8 | 8238 | e3a03000 | mov r3, #0 | Same as #4 |  |  |  |
| 9 | 823c | e50b3010 | str r3, [fp, #-16] | Same as #5 |  |  | offset\_12 = 0x10 |
| 10 | 8240 | e3a03000 | mov r3, #0 | Same as #4 |  |  |  |
| 11 | 8244 | e50b3014 | str r3, [fp, #-20] | Same as #5 |  |  | offset\_12 = 0x20 |
| **Assign gpio reg addr to pointers:** | | | | | | | |
| 12 | 8248 | e59f305c | ldr r3, [pc, #92] | <opcode>=0x59  Load word pre-idx  add imm offset,  r3<-addr[pc+92] |  |  | offset\_12 = 0x5c |
| 13 | 824c | e50b3008 | str r3, [fp, #-8] | Same as #5 | r3(0x3) | fp(0xb) | offset\_12 = 0x8 |
| 14 | 8250 | e59f3058 | ldr r3, [pc, #88] | Same as #12 | r3(0x3) | pc(0xf) | offset\_12 = 0x58 |
| 15 | 8254 | e50b300c | str r3, [fp, #-12] | Same as #5 | r3(0x3) | fp(0xb) | offset\_12 = 0xc |
| 16 | 8258 | e59f3054 | ldr r3, [pc, #84] | Same as #12 | r3(0x3) | pc(0xf) | offset\_12 = 0x54 |
| 17 | 825c | e50b3010 | str r3, [fp, #-16] | Same as #5 | r3(0x3) | fp(0xb) | offset\_12 = 0x10 |
| 18 | 8260 | e59f3050 | ldr r3, [pc, #80] | Same as #12 | r3(0x3) | pc(0xf) | offset\_12 = 0x50 |
| 19 | 8264 | e50b3014 | str r3, [fp, #-20] | Same as #5 | r3(0x3) | fp(0xb) | offset\_12 = 0x20 |
| 20 | 8268 | e51b3008 | ldr r3, [fp, #-8] | <opcode>=0x51  Load r2 word pre-idx  subtract imm offset  r3<-addr[fp-8] | r3(0x3) | fp(0xb) | offset\_12 = 0x8 |
| **Mask all but gpio pin 6:** | | | | | | | |
| 21 | 826c | e3e02040 | mvn r2, #64 | <opcode>=0x3e  mov not immdiate  r2<-!0x40 | r2(0x2) |  | imm\_8= 0x40  rotate\_imm=0 |
| 22 | 8270 | e5832000 | str r2, [r3] | <opcode>=0x58  Store word pre-idx  add imm offset  r2->addr [r3] | r2(0x2) | r3(0x3) | offset\_12 = 0x0 |
| **Make gpio pin 6 an output pin:** | | | | | | | |
| 23 | 8274 | e51b300c | ldr r3, [fp, #-12] | Same as #20 | r3(0x3) | fp(0xb) | offset\_12 = 0xc |
| 24 | 8278 | e5933000 | ldr r3, [r3] | Same as #12 | r3(0x3) | r3(0x3) | offset\_12 = 0x0 |
| 25 | 827c | e3832040 | orr r2, r3, #64 | <opcode>=0x38  OR immediate  r2<-r3|0x40 | r2(0x2) | r3(0x3) | imm\_8= 0x40  rotate\_imm=0 |
| 25 | 8280 | e51b300c | ldr r3, [fp, #-12] | Same as #20 | r3(0x3) | fp(0xb) | offset\_12 = 0xc |
| 26 | 8284 | e5832000 | str r2, [r3] | Same as #22 | r2(0x2) | r3(0x3) | offset\_12 = 0x0 |
| **Start of while loop :** | | | | | | | |
| 27 | 8288 | e51b2010 | ldr r2, [fp, #-16] | Same as #20 | r2(0x2) | fp(0xb) | offset\_12 = 0x10 |
| 28 | 828c | e5d23000 | ldrb r3, [r2] | <opcode>=0x5d  Load byte pre- idx add imm offset  r3<-byte at addr[r2] | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 29 | 8290 | e3833040 | orr r3, r3, #64 | Same as #25 | r3(0x3) | r3(0x3) | imm\_8= x40  rotate\_imm=0 |
| 30 | 8294 | e5c23000 | strb r3, [r2] | <opcode>=0x5c  Store byte pre- idx add imm offset  r3->byte at addr[r2] | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 31 | 8298 | e51b2010 | ldr r2, [fp, #-16] | Same as #20 | r2(0x2) | fp(0xb) | offset\_12 = 0x10 |
| 32 | 829c | e5d23000 | ldrb r3, [r2] | Same as #28 | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 33 | 82a0 | e3c33040 | bic r3, r3, #64 | <opcode>=0x3c  Bit Clear  r3<-r3&!0x40 | r3(0x3) | r3(0x3) | imm\_8= x40  rotate\_imm=0 |
| 34 | 82a4 | e5c23000 | strb r3, [r2] | Same as #30 | r3(0x3) | r2(0x2) | offset\_12 = 0x0 |
| 35 | 82a8 | eafffff6 | b 8288 | 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 | | |

# Hours:

4 hours : GPIO related reading, coding, makefile.  
10-12 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