

Lab 12: Airbag Trigger SOLUTIONS

Digital Design and Computer Architecture: RISC-V Edition (Harris & Harris, Elsevier © 2021)

Please indicate how many hours you spent on this lab. This will be helpful for calibrating the workload for next time the course is taught.

1. [3] Your assembly language implementation.

```
// lab9asm.c
// David_Harris@hmc.edu 29 March 2017
```

```
#include "EasyNucleoIO.h"
```

```
// prototype for asm file
```

```
void triggerCheck(void);
```

```
int main(void) {
    EasyNucleoIOInit();
    pinMode(0, INPUT);
    pinMode(1, INPUT);
    pinMode(2, OUTPUT);

    triggerCheck();
}
```

```
; triggerCheck.s
```

```
; David_Harris@hmc.edu 29 March 2017
```

```
; volatile unsigned long *GPIOA_IDR = (unsigned long*)0x48000010;
; volatile unsigned long *GPIOA_ODR = (unsigned long*)0x48000014;
; while(1) { // assume it is ok to set all other output bits to 0
;     if ((*GPIOA_IDR & INMASK) == INMASK) *GPIOA_ODR = D2MASK;
;     else *GPIOA_ODR = 0;
; }
```

```
AREA |.text|, CODE, READONLY
```

```
; define this file as code
```

```
EXPORT triggerCheck
```

```
; declare triggerCheck to be called externally
```

```
GPIOA_IDR EQU 0x48000010
```

```
; define constant address of port
```

```
GPIOA_ODR EQU 0x48000014
```

```
; define constant address of port
```

```
INMASK EQU 0x600
```

```
; 1 in bits 9 and 10 for PA9 and PA10 (D1, D0)
```

```
D2MASK EQU 0x1000
```

```
; 1 in bit 12 for PA12 (D2)
```

```
triggerCheck
```

```
LDR R0, =GPIOA_IDR
```

```
; put addresses in registers
```

```
LDR R1, =GPIOA_ODR
```

```
LDR R2, =INMASK
```

```
; put constants in registers
```

```
LDR R3, =D2MASK
```

```
LDR R4, =0
```

```
while
```

```
LDR R5, [R0]
```

```
; R5 = *GPIOA_IDR
```

```
ANDS R5, R2
```

```
; R5 = R5 & INMASK
```

```
CMP R5, R2
```

```
; R5 = INMASK? Both inputs true?
```

```
BNE else
```

```
STR R3, [R1]
```

```
; yes: *GPIOA_ODR = D2MASK
```

```
B while
```

```
else
```

```
STR R4, [R1]
```

```
; no: *GPIOA_ODR = 0
```

```
B while
```

```
ALIGN
```

```
; make sure code ends on word boundary
```

```
END
```

2. [3] Your optimized C implementation. (here is one possibility)

```
// lab9opt.c
// David_Harris@hmc.edu 29 March 2017

#include "EasyNucleoIO.h"

#define D0MASK (1 << 10)
#define D1MASK (1 << 9)
#define INMASK (D0MASK | D1MASK)
#define D2MASK (1 << 12)

void triggerCheck(void) {
    volatile unsigned long *GPIOA_IDR = (unsigned long*)0x48000010;
    volatile unsigned long *GPIOA_ODR = (unsigned long*)0x48000014;
    while(1) { // assume it is ok to set all other output bits to 0
        if ((*GPIOA_IDR & INMASK) == INMASK) *GPIOA_ODR = D2MASK;
        else *GPIOA_ODR = 0;
    }
}

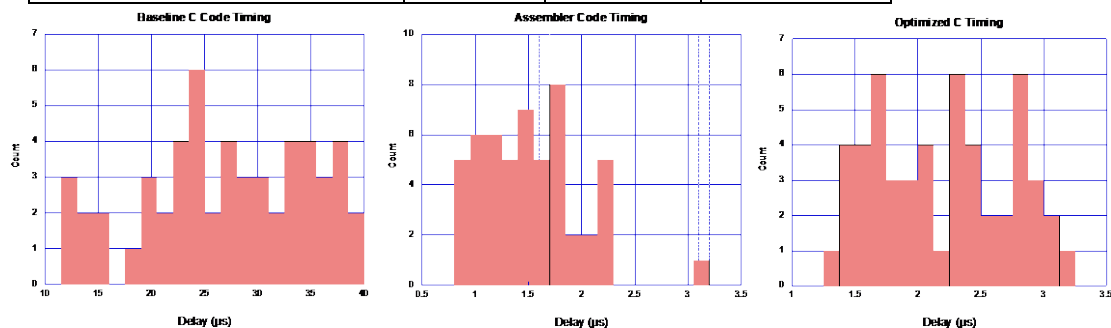
int main(void) {
    EasyNucleoIOInit();
    pinMode(0, INPUT);
    pinMode(1, INPUT);
    pinMode(2, OUTPUT);

    triggerCheck();
}
```

3. [3] A table of instruction count and average, max, and standard deviation of latency for each of the three implementations.

(these are my measurements. Yours will show some statistical variation, but should be in this general range)

	Baseline	Assembly	Optimized C
Greatest Instruction Count	130	6	10
Avg Latency (μ s)	27.072	1.515	2.197
Max Latency (μ s)	39.528	3.132	3.146
Stdev Latency (μ s)	11.988	0.830	1.329



The instruction count derivation is shown below.

Baseline

triggerCheck executes 20 instructions (excluding setup) + 2 calls to digitalRead and one to digitalWrite.

digitalRead executes 13 instructions + 1 call to pinToReg and 1 to pinToOffset.

digitalWrite executes 18 instructions + 1 call to pinToReg and 1 to pinToOffset.

pinToReg executes 10 instructions (to return PORTB).

pinToOffset executes 12 instructions.

Hence, triggerCheck executes $20 + 2 * (13 + 10 + 12) + (18 + 10 + 12) = 130$ instructions.

Assembly

The while loop executes 6 instructions.

```
26: while
0x0800016C 4C08      LDR      r4,[pc,#32] ; @0x08000190
27:                LDR R5, [R0] ; R5 = *GPIOA_IDR
0x0800016E 6805      LDR      r5,[r0,#0x00]
28:                ANDS R5, R2 ; R5 = R5 & INMASK
0x08000170 4015      ANDS      r5,r5,r2
29:                CMP R5, R2 ; R5 = INMASK? Both inputs true?
0x08000172 4295      CMP      r5,r2
30:                BNE else
0x08000174 D101      BNE      0x0800017A
31:                STR R3, [R1] ; yes: *GPIO_ODR = D2MASK
0x08000176 600B      STR      r3,[r1,#0x00]
32:                B while
33: else
0x08000178 E7F9      B        0x0800016E
34:                STR R5, [R1] ; no: *GPIO_ODR = 0
0x0800017A 600D      STR      r5,[r1,#0x00]
35:                B while
0x0800017C E7F7      B        0x0800016E
```

Optimized C

The while loop executes 10 instructions in the worst case.

```
13:                while(1) { // assume it is ok to set all other output bits
to 0
0x08000374 E00B      B        0x0800038E
14:                if ((*GPIOA_IDR & INMASK) == INMASK) *GPIOA_ODR = D2MASK;
0x08000376 680A      LDR      r2,[r1,#0x00]
0x08000378 2303      MOVS     r3,#0x03
0x0800037A 025B      LSLS     r3,r3,#9
0x0800037C 401A      ANDS     r2,r2,r3
0x0800037E 429A      CMP      r2,r3
0x08000380 D103      BNE      0x0800038A
0x08000382 2201      MOVS     r2,#0x01
0x08000384 0312      LSLS     r2,r2,#12
0x08000386 6002      STR      r2,[r0,#0x00]
```

0x08000388	E001	B	0x0800038E
15:	else		*GPIOA_ODR = 0;
0x0800038A	2200	MOVS	r2, #0x00
0x0800038C	6002	STR	r2, [r0, #0x00]
0x0800038E	E7F2	B	0x08000376

If you have suggestions for further improvements of this lab, you're welcome to include them at the end of your lab.