

BLG212E-Homework 2

Altay Ünal: unal21@itu.edu.tr

31 December 2021

1 Introduction

In this homework, you are expected to implement some functions with Assembly using interrupts and GPIO.

2 Part 1-Pooling

In convolutional neural networks, an input image is fed into the network and the features of that image are learnt through the network. During learning phase, convolutional and pooling layers are used to extract features of the image. Pooling layers reduces the dimensions and decrease the number of computations and computation time significantly. There are two types of pooling, maximum pooling and average pooling, respectively. In maximum pooling, the input matrix is checked through filters and the maximum element in that filter is written to the output matrix. In average pooling, the average of the elements in the filter is computed and written to the output matrix.

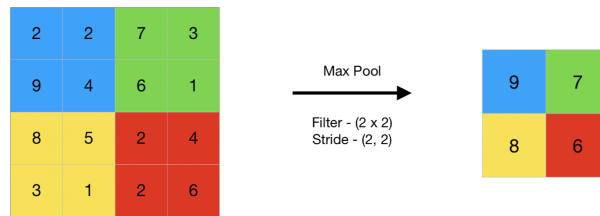


Figure 1: Max Pooling with 2x2 filter.

In this homework, you are expected to implement the given pooling mechanisms. You are given 4 arrays with 10 elements which are representing the rows of the matrix respectively. Your filter size is 2x2 and your stride is (2,2) meaning that after you applied pooling to a section, you must go 2 rows forward similar to Figure-1. You should use SysTick as timer and your filtering operation must occur during the SysTick interrupt. After reading the arrays and activating the

SysTick timer, you pass your filter through the arrays, determine the maximum element in the filter and write your determined value to the memory. At the end of the operation, you should obtain a 2x5 matrix with the maximum values obtained from the filters.

The process for the average pooling is similar to the maximum pooling. But this time, the average of the elements must be computed and written to the memory. Be careful that **all filtering processes must be done in interrupt.** In your main code, you can only write your interrupt outputs to the memory.

The activation of the SysTick timer in Startup file is given below. You basically need to comment the default handler and enter the name of your SysTick interrupt service routine. **Do not forget to export your interrupt service routine so that your interrupt can be handled without any issues.**

```

SysTick_Handler PROC
    EXPORT SysTick_Handler      [WEAK]
    B      SysTickIsr
    ENDP

; Macro to define default exception/interrupt handlers.
; Default handler are weak symbols with an endless loop.
; They can be overwritten by real handlers.
MACRO
    Set_Default_Handler $Handler_Name
$Handler_Name PROC
    EXPORT $Handler_Name      [WEAK]
    B      .
    ENDP
MEND

; Default exception/interrupt handler

Set_Default_Handler NMI_Handler
Set_Default_Handler SVC_Handler
Set_Default_Handler PendSV_Handler
;Set_Default_Handler SysTick_Handler

```

Figure 2: SysTick configuration in Startup file.

IMPORTANT NOTE: The registers for SysTick timer are given below. You must configure these registers first.

Address	Name	Type	Required privilege	Reset value	Description
0xE000E010	SYST_CSR	RW	Privileged	0x00000000	<i>SysTick Control and Status Register.</i>
0xE000E014	SYST_RVR	RW	Privileged	Unknown	<i>SysTick Reload Value Register on page 4-17.</i>
0xE000E018	SYST_CVR	RW	Privileged	Unknown	<i>SysTick Current Value Register on page 4-18.</i>

Figure 3: SysTick registers.

3 Part 2-Blinking LED

IMPORTANT NOTE: For this part, you need to use MKL25Z4 instead of ARM Cortex M0.

In this part, you need to write an Assembly code to blink LED on the KL25Z board. The period of LED must be 1 second initially and if the switch is pressed once, the period becomes 2 seconds. If the switch is pressed for the second time, the period becomes 4 seconds and if the switch is pressed for the third time, the period must be reset. The registers of the GPIO in KL25Z board is provided in your lecture notes.

While implementing your program, you must consider the debouncing case for the GPIO. Debouncing is called for the case where the input from GPIO is pulled high (or low) quickly at very short interval due to mechanical reasons. Although, you run your programs in a simulator, this is an important problem for GPIOs and you must solve this issue in your programs.

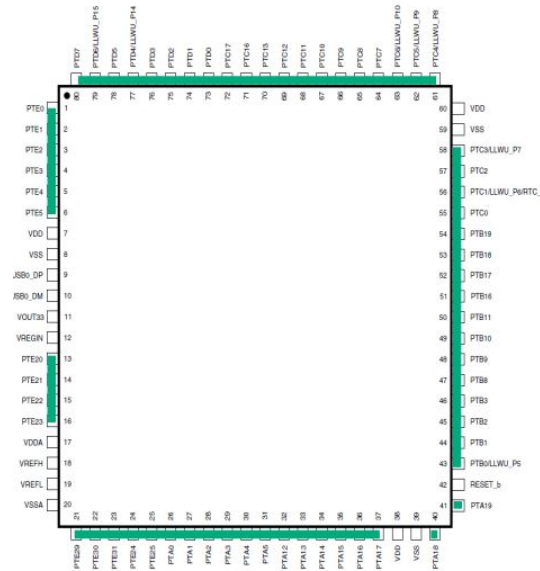


Figure 4: KL25Z and GPIO ports.

Absolute address (hex)	Register name	Width (in bits)
400F_F000	Port Data Output Register (GPIOA_ODOR)	32
400F_F004	Port Set Output Register (GPIOA_OSOR)	32
400F_F008	Port Clear Output Register (GPIOA_OCOR)	32
400F_F00C	Port Toggle Output Register (GPIOA_OTOR)	32
400F_F010	Port Data Input Register (GPIOA_ODIR)	32
400F_F014	Port Data Direction Register (GPIOA_ODDR)	32

Figure 5: GPIO registers for PORTA.