Datorteknik Laboration 3

Högskolan i Halmstad
André Frisk & Fredrik Kortetjärvi

Uppgift 1

Fråga 1: Vad händer med stacken vid avbrott och vad sparas undan?
SVAR: Programmet sparar registren från R0-R3, R12, LR (R14), PC och SP (R13). Den läser av adressen från avbrottshanteraren.

Fråga 2 Vad står i LR-registret när avbrottshanteraren körs, och vad betyder det?
SVAR: Adressen för där avbrottet aktiverades. När avbrottet avslutas returnerar programmet en till den positionen.

Fråga 3: Vilken instruktion gör att avbrottet avslutas, dvs hur återgår ni till huvudprogrammet?
SVAR: BX LR

Fråga 4: Var lagras återhoppsadressen?
SVAR: Det lagras i LR registret.

Uppgift 2

Fråga 1: Vilken adress har NVIC-registret ISER0?
SVAR: 0xE000E100

Fråga 2: Hur behöver ni konfigurera PIOA för att få knappen att reagera bara på negativ flank?
SVAR: Ett register måste laddas med registret från PIOA_FELLSR och därefter lagras i ett annat register. Ex. LDR R0, =PIOA_FELLSR
STR R1, [R0]

PIOA_FELLSR används vid falling edge eller på svenska som det heter ”negativ flank”.

Fråga 3: Vad händer om man får knappavbrott då man stegar genom koden? Testa genom att trycka på USR_LEFT när du stegar genom avbrottskoden. Samtidigt kolla NVIC-registret ISPR0 (eller ICPR0) med View Memory. Vad händer?
SVAR: Det har med bitmasken att göra. Den läser av bitmasken och då ser ISPR0 om knappen är nedtryckt, då får den värdet i knapptrycket vilket är 8 (startvärdet för lampan).

Uppgift 3

```
; Initialization InPort
  LDR R0, =PIOA_PER
  MOV R1, #3<<14       ; enable PIOA on bit 14 and 15
  STR R1, [R0]
  LDR R0, =PIOA_ODR
  STR R1, [R0]         ; set bit 15 and 14 as inport
  LDR R0, =PIOA_PUER
  STR R1, [R0]
  MOV R1, #1<<14
  LDR R0, =PIOA_IER
  STR R1, [R0]
  LDR R0, =PIOA_FELLSR
  STR R1, [R0]
  LDR R0, =PIOA_AIMER
  STR R1, [R0]
  LDR R0, =ICPR
  MOV R1, #1<<11
  STR R1, [R0]
  LDR R0, =ISER
  STR R1, [R0]

  LDR R0, =PIOC_PER
  MOV R1, #14
  STR R1, [R0]
  LDR R0, =PIOC_OER
  STR R1, [R0]
  LDR R0, =PIOC_PUDR
  STR R1, [R0]
```

```
  LDR R0, =SYSTICK_CTRL
  LDR R1, =0
  STR R1, [R0]
  LDR R0, =SYSTICK_LOAD
  LDR R1, =0xB71AFF
  STR R1, [R0]
  LDR R0, =SYSTICK_VAL
  LDR R1, =0
  STR R1, [R0]
  LDR R0, =SYSTICK_CTRL
  LDR R1, =7
  STR R1, [R0]
```

```
; Main
 MOV R7, #500          ;Delay
Loop
 LDR R0, =PIOA_PDSR
 LDR R1, [R0]
 AND R1, R1, #0x8000     ; mask left button
 CMP R1, #0
 BNE Loop

 LDR R0, =PIOA_IDR
 LDR R2, =0x8000
 STR R2, [R0]

 LDR R0, =PIOC_PDSR
 LDR R1, [R0]
 AND R1, R1, #6          ; mask bit 2 and 1
 LSR R1, R1, #1          ; move to bit 1 and 0
 ADD R1, R1, #1          ; add LEDs +1
 LSL R1, R1, #1          ; shift bits to align for LEDs

 LDR R0, =PIOC_SODR
 STR R1, [R0]
 EOR R1, R1, #6          ; invert bit 2 and 1
 LDR R0, =PIOC_CODR      ; address to Clear Output Data Register
 STR R1, [R0]
 LDR R0, =PIOA_IER       ; address to Interrupt Enable Register
 STR R2, [R0]            ; enable Interrupt again
```

```
main_release
 LDR R0, =PIOA_PDSR
 LDR R1, [R0]
 AND R1, R1 , #1<<15
 LSR R1, R1, #15
 CMP R1, #1
 BNE  main_release
 BL Delay_ms
 B  main_loop


;Subrutin Delay_ms
;----------------------------------------
; Vänta ett antal ms
; Inparameter: R7 -delay i ms
; -----------------------------------------------
DELAY_CALIB EQU 1200
; utprovat värde (baserat på master clock -12Mhz)
Delay_ms
  STMFD SP!,{R0,R1}
  MOV  R0,R7
do_delay_ms
  LDR  R1,=DELAY_CALIB
loop_ms
  SUBS R1,R1,#1
  BNE  loop_ms
  SUBS R0,R0,#1
  BNE  do_delay_ms
  LDMFD SP!,{R0,R1}
  BX LR
```

```
SysTick_Handler
  LDR R0, =PIOC_PDSR
  LDR R1, [R0]
  AND R1, R1, #8            ; mask 8 (LED on bit 3)
  LDR R0, =PIOC_CODR        ; toggle LED 3
  STR R1, [R0]
  EOR R1, R1, #8
  LDR R0, =PIOC_SODR
  STR R1, [R0]
  BX LR

PIOA_Handler
  LDR R0, =PIOA_ISR
  LDR R1, [R0]             ; read Interrupt Status Register
  AND R1, R1, #1<<14       ; mask bit 14 (left button)
  LSR R1, R1, #14          ; move to right
  CMP R1, #1
  BNE PIOA_Handler_Exit
  LDR R0, =PIOA_PDSR
  LDR R1, [R0]
  AND R1, R1, #1<<15       ; mask bit 15 (right button)
  CMP R1, #0
  BNE PIOA_Handler_Set
  LDR R0, =PIOC_CODR        ; reset by turning off LEDs and set systick_load to default
  MOV R1, #14
  STR R1, [R0]
  LDR R0, =SYSTICK_LOAD
  LDR R1, =0xB71AFF
  STR R1, [R0]
  B PIOA_Handler_Exit
```

```
PIOA_Handler_Set           ;new freq
  LDR R0, =PIOC_PDSR
  LDR R1, [R0]
  AND R1, R1, #6
  LSR R1, R1, #1
  LDR R2, =0xB71B00
  LSR R2, R2, R1
  SUB R2, R2, #1
  LDR R0, =SYSTICK_LOAD
  STR R2, [R0]

PIOA_Handler_Exit
  BX LR

  END
```

Programmet börjar med att skriva in initieringsadresserna för PIOA, PIOC, PMC, SYSTICK, NVIC. I Initialization InPort initieras alla register såsom interrupts, systick, pin enable etc.

I Main ges värdet av #500 till R7 som står för delay vid uppstart av programmet. Alla pins initieras med R0 och R1 lagras med adressen för initiering av pinsen. Programmet kommer att stanna inom Loop sålänge en knapp inte är nedtryckt. Inuti Loop kollar programmet med maskning om det specifikt är left-button som är nedtryckt och använder CMP för att se om den är nedtryckt (#0). Vid nedtryckt vänsterknapp går programmet ut ur loopen.

R0 får adressen till att avbryta och lagras i R2. Programmet laddar in adressen för lamporna i R0 och laddas till R1. I R1 tas bitarna 1 och 2 ut, och flyttar de till 0 respektive 1 (Logical Shift Right, flyttas ett steg till höger) och adderar 1 för att få det jämnt/ojämnt för lampans tändning och släckning. Bitarna flyttas tillbaka till sin ursprungsposition med hjälp av LSL (Logical Shift Left, flyttas ett steg till vänster).

R0 laddas med adressen från registret med att starta en lamporna beroende på vilken lampa som är igång som beror på maskningen i föregående stycke, som lagras i R1. R1 inverterar bit 1 och 2 för att göra tvärtom av det som hände vid adderingen av 1 i föregående stycke. R0 får adressen inladdad för att göra en interrupt och lagras i R2.

Om en interrupt sker utav PIOA skicka man till PIOA_Handler. R0 får adressen för att se om det blev en interrupt eller inte som laddas in till R1. Programmet kollar biten som ändras när vi trycker ner på left och vi flyttar den till första värdet till höger. Jämförelse på R1 om vänstra knappen är nedtryckt, om den inte är lika med skickar programmet en till PIOA_Handler_Exit som därefter skickar en till main_release (BX LR). Annars läser programmet om den högra knappen är nedtryckt och kollar samma steg som på vänstra. Om knappen är nedtryckt skickas man till PIOA_Handler_Set (skrivs om i nästa stycke). Om inte rensas registrena och systick_load går till default vilket är 0 och stänger av lampan. Vid slutet på PIOA_Handler skickar programmet en till PIOA_Handler_Exit som skickar en till main_release.

På PIOA_Handler_Set sätter den nya frekvensen. Adressen för pinen skickas till R0 och laddas in till R1. R1 right-shiftar ett steg till höger. R2 laddas med ett nytt värde för att få en snabbare frekvens på tändning och släckning av lampan. R2 right-shiftar med R1 för att få frekvensen att bli snabbare och laddar R2 med SYSTICK_LOAD (R0).

Om det inte blir en interrupt går programmet in i main_release och fortsätter i en loop där mittenlampan tänder och släcker hela tiden (i SysTick_Handler) och delayn sker. Detta tills att en interrupt initieras.