Datorteknik Laboration 2

Högskolan i Halmstad
André Frisk & Fredrik Kortetjärvi

Uppgift 1

Leta reda på de olika registrens offsettadresser ():

- PIOA_PER; 0x400E0E00
- PIOA_ODR; 0x400E0E14
- PIOA_IDR; 0x400E0E44
- PIOA_PUER; 0x400E0E64
- PIOA_PDSR; 0x400E0E3C

Memory-uppgifter:

Fråga 1: Vilken bit är det som ändras?
SVAR: bit 15

Fråga 2: Gör samma sak med den högra knappen. Vilken bit ändras då?
SVAR: bit 16

Fråga 3: Är biten 1 eller 0 då knappen är nedtryckt?
SVAR: 0

Fråga 4: Vilken programrad är det som gör själva inläsningen till minnet?
SVAR: LDR R1, =PIOA_PDSR

Kod för asm1:

Koden som visas är själva koden som använda för att tända LED. Adresserna för varje Register finns i asm1 filen.

Hur fungerar programmet?

```
main      LDR   R0,=PMC_PCER
          LDR   R1,=(5 << 11)  ; Peripheral Identifier: bit11=PIOA, bit13=PIOC
          STR   R1,[R0]
          LDR R1,=PIOC_PER ;enable the pins we will use output
          MOV R2, #6
          STR R2, [R1]
          LDR R1,=PIOC_OER ; Output Enable Register
          MOV R2, #6
          STR R2, [R1]
          LDR R1,=PIOC_PUDR  ; Pull Up Disable Register
          MOV R2, #6
          STR R2, [R1]
```

I denna kodsektion initieras själva pinsen som skall användas till uppgiften för att sätta igång lamporna med output. R0 får värdet av en klocka med värdet 0 vilket betyder att lampan kommer vara statisk vid tändning.

```
LOOP
  LDR R1,=PIOA_PDSR ; Adress till PortA:s dataregister
  LDR R0,[R1] ; Läser från adress i R1 till R0
  AND R1, R0, #0X4000 ; R0 AND:as med
  AND R2, R0, #0X8000 ; R0 AND:as med
  CMP R1, #0
  BEQ   Left
  CMP R2, #0
  BEQ   Right
  B LOOP ; En evig loop
Left
  MOV R2, #2
  LDR R1, =PIOC_SODR ;turn on the leds

  STR R2, [R1]
  LDR R1,=PIOA_PDSR ; Adress till PortA:s dataregister
  LDR R0,[R1] ; Läser från adress i R1 till R0
  AND R2, R0, #0X8000 ; R0 AND:as med
  CMP R2, #0
  BEQ   Both
  B Left
Right
  MOV R2, #4
  LDR R1, =PIOC_SODR ;turn on the leds
```

```
  STR R2, [R1]
  LDR R1,=PIOA_PDSR ; Adress till PortA:s dataregister
  LDR R0,[R1] ; Läser från adress i R1 till R0
  AND R2, R0, #0X4000 ; R0 AND:as med
  CMP R2, #0
  BEQ   Both
  B Right
Both
  MOV R2, #6
  LDR R1, =PIOC_SODR ;turn on the leds
  STR R2,[R1]

  B Both
```

PortA initieras med att man tar adressen för PortA till R1. R0 får adressen från R1 då man måste gå denna omväg då det inte går att läsa direkt från PortA, gäller alltid vid assemblerprogrammering. R0 AND:as med en bitmask med R1 (portvärde) och R2 (portvärde) för att plocka ut information från en enskild bit (knapptryck). Detta för att datorn behöver denna bit till programmet för att det ska vara funktionellt.

Vid denna punkt i programmet vid CMP radera bör vänstra eller högra knappen vara intryckta. Om vänster är intryck (R1) ska programmet förflytta en till Left. Om höger är intryckt (R2) ska programmet förflytta en till Right. Programmet går in i en evig loop tills ett av villkoren har uppnåtts.

I Left får R2 värdet #2 då detta är initieringsvärdet för att den vänstra lampan ska tändas. R1 laddas med adressen för att tända lampan. R2 lagras med värdet från R1 (knapptrycket för att starta lampan). Nu borde lampan vara tänd. Adressen från PortA skickas in i R1 som därefter laddas till R0 där maskning sker med R2 för att få ut denna specifika biten av information (knapptryck). Programmet jämför om den andra knappen (i detta fall höger) är nertryckt (värdet 0) med R2, är

båda nertryckta skickar programmet en till Both. Annars går programmet runt i Left infinitivt för att hålla lampan tänd.

I Right händer exakt det som händer i Left förutom att R2 får värdet #4 då detta är initieringsvärdet för att den högra lampan ska tändas. Ett annat värde för maskningen används med. Om den vänstra knappen är nedtryckt inne i Right skickas man till Both, annars är den högra lampan igång hela tiden.

I Both får R2 värdet #6 som är initieringsvärdet för att alla lampor ska tändas (vänster och höger). R1 får adressen för att sätta igång lamporna och därefter lagras just den adressen i R2. Vid branchen hoppar programmet tillbaka till Both och körs infinitivt. Lamporna är därefter alltid tända.

Uppgift 2

Då koden för uppgift 2 är väldigt lik koden i uppgift 3 förklaras uppgift 2 koden i uppgift 3. Enbart frågorna kommer att besvaras i denna uppgift.

Fråga 1: Vad är skillnaden mellan Step Over och Step Into?
SVAR: Step Into hoppar in och visar alla steg i subrutinen medans Step Over går igenom subrutinen men visar inte stegen inuti.

Fråga 2: Vilket värde har stackpekaren (R13)?
SVAR: 0x20008000

Fråga 3: Vilket värde har länkregistret (R14)?
SVAR: 0x200702F5

Fråga 4: Vilken adress sparas i R0 (vid stegning för stacken)?
SVAR: 0x20007FFC

Fråga 5: Vilken adress sparas i R1 (vid stegning för stacken)?
SVAR: 0x20007FF8

Fråga 6: Åt vilket håll växer stacken?
SVAR: Neråt, 4 steg

Fråga 7: Vilket värde laddas programräknaren med (subrutin)?
SVAR: 0x20070308

Fråga 8: Till vilken adress hoppar programmet (subrutin)?
SVAR: 0x200702F4 (slutet av Loop)

Uppgift 3

I denna uppgift ska programmet släcka och tända lamporna. Notera att i koden är adresserna för de olika Register identiska som de i uppgift 1. Även att koden börjar på samma sätt som första bilden på uppgift 1 där alla pins blir aktiva, den enda skillnaden är att istället för att ge R2 värdet #6 är det #14. Det beror på att det är initieringsvärdet för att alla lampor ska tändas som i detta fall är tre lampor.

Följande kod tillhör uppgift 3 (se nästa sida):

```
; Huvudprogram -------------------------------------
read
      BL LED_p0
      B read

Loop  B read
; ---------------------------------------------------------------
STOP    B STOP

;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
```

```
; ---------------------------------------------------------
;Subrutin Read_p0 ----------------------------------------
; Avläsning av knapp i subrutin
; Utparametrar: R0
; ---------------------------------------------------------
LED_p0
  STMFD SP!,{R1}
  MOV R5, R14
  MOV R2, #14
  LDR R1, =PIOC_SODR ;turn on the leds
  STR R2, [R1]
  MOV R7, #1000
  BL Delay_ms
  LDR R1, =PIOC_CODR ;turn off the leds
  STR R2, [R1]
  LDR R0,[R1]
  MOV R7, #1000
  BL Delay_ms
  LDMFD SP!,{R1}
  BX R5
; ---------------------------------------------------------

        END
```

Hur fungerar programmet?

Programmet börjar med att direkt hoppa till subrutinen LED_p0. R1 registret sparas till stacken på första kodraden. R5 får värdet från R14 för att kunna ta sig vidare i programmet senare (felaktiga funktioner uppstod annars). R2 får värdet #14 till initiering av att sätta igång alla lampor. R1 får adressen till att sätta igång lamporna och i R2 lagras värdet av R1 (måste göras i assembly för att

fungera). R7 får värdet #1000 vilket i detta fall betyder en sekund. Detta värde skickas in till subrutinen Delay_ms.

I Delay_ms sker själva fördröjningen mellan tändningen och släckningen i programmet där R7 som bestämdes i förväg är antal millisekunder som det ska fördröjas. Den inre loopen tar precis 1 ms att exekvera och den yttre räknar upp till det tal som är i R7 alltså fördröjningen. Subrutinen använder internt R0 och R1 som sparas i stacken och återställs i slutet av subrutinen för att inte skapa problem för programmet om den kommer in i Delay_ms igen. När subrutinen har kommit upp till fördröjningsvärdet skickas det tillbaka till LED_p0 (raden efter BL Delay_ms).

Där får R1 adressen för att stänga av lamporna och värdet lagras i R2. Här följer samma procedur igen med att R7 ges värdet för en sekunds fördröjning och skickas in i Delay_ms. När fördröjningen är slutförd skickar programmet oss tillbaka till Led_p0 och R1 återställs från stacken. På BX R5 skickar programmet till Loop och kör om programmet infinitivt så att programmet sätter på och stänger av lamporna med en fördröjning som användaren själv har bestämt.