TSEA28 Datorteknik Y, lösningar till tentamen 23-05-30

1. a) Beräkna PM(ASR)-GRx. Om N-flaggan = O-flaggan är GRx <= PM(ASR) och då är instruktionen klar. Annars räkna upp PC med 2 och avsluta instruktionen.

```
Adress Mikrokod                              Kommentar
0x0b   buss:=PM, AR:=buss, uPC:=uPC+1        ; Läs PM till AR
0x0c   buss:=GRx, R:=0, S:=0, AR:=AR-buss,   ; Beräkna PM-Grx, påverkar flaggor
       uPC:=uPC+1                            ;
0x0d   uPC:=0x11 om N=1 annars uPC:=uPC+1    ; N-flaggan testas
0x0e   uPC:=0x0 om O=0 annars uPC:=uPC+1     ; N=0, nästa instruktion om O=0
0x0f   PC:=PC+1, uPC:=uPC+1                  ; N=0,O=1, Öka PC med 1
0x10   PC:=PC+1, uPC:=0                      ; Ökat PC med 2, nästa instruktion
0x11   uPC:=0x0 om O=1 annars uPC:=uPC+1     ; N=1, nästa instruktion om O=1
0x12   PC:=PC+1, uPC:=0x10                   ; N=1, O=0, Öka PC med 1
```

b) Index 6 => Opcode = 0110, GRx=01, M=10, A=0x5a => maskininstruktion = opcode GRx M A = 0110 0110 0101 1010_2 = 0x665a

2. a) SRAM är snabbare, dyrare per bit, större area per bit, enklare att kommunicera med än DRAM (dvs DRAM periodisk refresh av minnesinnehållet). DRAM kan läsa ut data i burst. SRAM drar mer energi än DRAM vis motsvarande datahastigheter (dock kan SRAM stoppas och drar då mycket mindre energi).

b) Nej, data forwarding hjälper till med datakonflikter. Styrkonflikter beror på avsaknad av information om att det är ett hopp och/eller att flaggorna från en beräkning inte beräknats än. Data forwarding kan hjälpa till med att få beräkningen att ske fortare, men då är det en datakonflikt som löses, inte en styrkonflikt.

c) En fullt associativ cache kan lägga data från en specifik adress på vilken cacheline som helst. Därför måste tag jämföras med alla taglines samtidigt vilket kräver lika många jämförelser som det finns cachelines. I en direktadresserad cache får data bara placeras på en specifik cacheline, så då räcker det med en jämförelse mellan lagrad tag och tagdel hos aktuell adress.

Eftersom fler adressbitar används för att peka på en specifik cacheline i en direktadresserad cache jämfört med en fullt associativ cache så kommer tag även bli längre i en fullt associativ cache jämfört med en direktadresserad cache. Så mängden minne för lagring av tag är större i den fullt associativa cachen.

d) Nej, eftersom N=1 betyder att MSB i registret är 1, och Z=1 betyder att alla bitar i register (inklusive MSB) =0. MSB kan inte både vara 0 och 1.

e) 2-vägs superskalär processor kan som mest starta 2 instruktioner per klockcykel => 2*3 miljoner instruktioner per sekund. 7-stegs pipelined processor kan maximalt starta 1 instruktion per klockcykel => 5 miljoner instruktioner per sekund. Eftersom 6 > 5 => 2-vägs superskalär processor kan utföra flest instruktioner per sekund.

TSEA28 Datorteknik Y, lösningar till tentamen 23-05-30

3. Programidé: gå igenom rabell 1 rad för rad. Använd värdet för att beräkna vilken adress indexet in i tabell 2 betyder, och lägg till värdet till befintligt värde i tabell 2.

```
subrutin:   ldr     r3,[r0],#4      ; läs en rad, låt r0 peka på nästa
            lsr     r4,r3,#(28-2)   ; Ta fram index och beräkna antal byte in i tabell2
                                    ; Skifta ned 28 bitpositioner (=> värde 0-15),
                                    ; multiplicera sedan med 4 (dvs skift vänster 2 steg)
            mvn r5,#0xf0000000      ; sätt r5 till 0x0fffffff (kan även använda mov/movt)
            and     r3,r5           ; nollställ index-bitarnas värde
            ldr     r5,[r2,r4]      ; läs värde ur tabell2
            add     r5,r3           ; Summera med värde från tabell 1
            str     r5,[r2,r4]      ; skriv tillbaks till tabell 2
            subs    r1,#1           ; en rad klar, är det fler kvar?
            bne     subrutin        ; ja, hoppa tillbaks och ta nästa
            bx      lr
```

4. Avbrottsrutin så alla generella register måste återställas efter anrop. För ARM Cortex-M görs detta automatiskt för R0-R3 och R12. Använd ldrh respektive strb för att bara läsa/skriva endast det antal byte som angetts i uppgiften.

```
IOPORT          .equ 0x40001008

avbrott:    cpsid   i           ; se till att inga andra avbrott kan ske
            push    {lr,r5,r6}  ; måste spara undan register eftersom det är avbrott
            mov     r3,#(IOPORT & 0xffff)   ; Peka på I/O port
            movt    r3,#(IOPORT >> 16)
            ldrh    r0,[r3]     ; Läs alla 16 bitar bara en gång (dvs 2 byte)
            mov     r1,r0
            eor     r0,#0x0f    ; invertera bit 3 till bit 0.
            mov     r2,#0x0123  ; lång konstant, måste använda register
            cmp     r0,r2       ; jämför förändrat värde med konstant
            ble     done        ; hoppa förbi om mindre eller lika med
            bl      Subrutin1
done:       strb    r1,[r3,#8]  ; skriv en byte till ut adress 0x40001010
            pop     {lr,r5,r6}  ; återställ register (r0-r3 fixas automatiskt)
            cpsie   i
            bx      lr
```

5. a) 0x86+0x7a = 0x10, dvs carry ut och värdet 0x00 i resultatregistret (8-bitar register då det är en 8-bitars processor). Därför blir Z=1, C=1. Eftersom 0x86 är negativt och 0x7a positivt om värdena ses som 2-komplementstal så kommer summan alltid få plats i resultatregistret. Därför är O=0. N=0 eftersom MSB i resultatregistret är 0.

b) Måste först teckenförlänga det kortaste talet till 5 bitar. Eftersom båda tal negativa måste resultat också bli negativt (behöver antingen öka till 6 bitar på båda eller kontrollera att additionen gav negativt svar).

```
Carry:  11111
         10111
        +11011
         -----
         10010
```

Svar: 10010_{2C}

Kontrollräkna gärna: 10111_{2C}= -16+4+2+1=-9, 1011_{2C}=-5, svaret är -16 + 2 = -14 vilket stämmer med -9 +(-5) = -9 -5 = -14.

c) -23 = - (23) = - (16+4+2+1) = - (00010111_2) = 11101000+1 = 11101001_{2C} där bitsekvensen även kan anges i hexadecimal form 0xE9.

6. a) Med 7 bitar index blir det 128 cachelines per väg. Med 4 vägar och 64 byte per cacheline blir den totala storleken på cachen 128*4*64 = 32*1024 = 32768 byte (eller 32 KiB).

b) 64 byte per cacheline => byteposition behöver 6 bitar (2^6 = 64). Antal bitar i adressen som återstår efter index och byteposition allokerats blir 32-7-6=19 bitar

c) De adresser som ska läsas är sekvensen nedan, med tag (19 bit), index (7 bit) och byteposition (6 bit) indikerat

0x12345678: tag=0x091a2, index=0x59, byteposition=0x38
0x12346e78: tag=0x091a3, index=0x39, byteposition=0x38
0x12348678: tag=0x091a4, index=0x19, byteposition=0x38
0x12349e78: tag=0x091a4, index=0x79, byteposition=0x38
0x1234b678: tag=0x091a5, index=0x59, byteposition=0x38
0x1234ce78: tag=0x091a6, index=0x39, byteposition=0x38
:

Index repeteras efter 4 läsningar. Adresserna ger aldrig både index och tag med samma värde, så en ny cacheline används för varje läsning. Då det finns 4 vägar så dessa 4 index kan repeteras totalt 4*4 gånger innan någon behöver kastas ut ur cachen. Svar: 16 adresser kan läsas innan något kastas ut ur cachen.

Notera att det kan vara lättare att se mönstret om man skriver adresserna i binär form.