TSEA28 Datorteknik Y, lösningar till tentamen 161018, reviderad 171020

1. a) Algoritm: flytta GR3 till AR, flytta GR3 till ASR, läs PM till PC, minska AR med 1, spara i GR3.

```
Adress Mikrokod                                  Kommentar
  11   buss:=GRx, R:=1, S:=0, AR:=buss,         ; flytta GR3 till AR (för att
       uPC:=uPC+1                                 kunna minska stackpekare)
  12   buss:=GRx, R:=1, S:=0, ASR=buss,         ; flytta GR3 till ASR (för att
       uPC:=uPC+1                               ; läsa stackens innehåll i PM)
  13   buss:=PM, R:=0, S:=0, PC:=buss,          ; flytta PM till PC (för att
       uPC:=uPC+1                               ; sätta PC till återhoppsadress)
  14   AR:=AR+buss, buss:=um (um=1)             ; Addera 1 (i mikrominnet) till AR
                                                ; automatiskt utförs uPC:=uPC+1
  15   buss:=AR, R:=1, S:=0, Grx:=buss,         ; flytta AR till GRx, ta nästa
       uPC:=0                                   ; instruktion via hopp till 0
```

b) Felaktig beskrivning i uppgiften: Saknar opcode för STORE, och det ska endast vara en kodning för JNE. Antar här att sista opkode ska vara STORE.

Mikrominnet har 128 adresser, därför ska 7 bitar finnas för mikrominnesadresserna. 4 opcodebitar ger 16 olika opcoder, och M1 är därför 16 adresser med 7 bitar per adress i M1.

```
Adress Mikrokodsadress          Adress Mikrokodsadress
 0000   -------                 1000   -------
 0001   0001011  ;RTS           1001   -------
 0010   -------                 1010   -------
 0011   -------                 1011   -------
 0100   -------                 1100   0001010  ;LOAD
 0101   0001000  ;JNE           1101   0000111  ;STORE
 0110   -------                 1110   -------
 0111   -------                 1111   -------
```

c) I mikrokoden utförs följande adresser vid RTS: 0, 1, 2, 3, 11, 12, 13, 14, 15. Dvs det är 9 steg vilket tar 9 klockcykler.

2. a) SRAM och DRAM tappar värdet när strömmen försvinner.

b) SRAM används i cacheminnet därför det är snabbt.

c) Negativt resultat indikeras av N-flaggan (negativt resultat) och O-flaggan (spillflagga, ibland kallad V). Om N=1 och O=0 eller om N=0 och O=1 har resultatet blivit negativt.

d) En SIMD-processor (Single Instruction Multiple Data) utför samma instruktion på många data samtidigt medan en vanlig processor utför en instruktion på ett data per gång.

e) I en pipelinad processor har inte föregående instruktion utförts när nästa instruktion ska hämtas. Ett villkorligt hopp har därför inte tillgång till flaggans värde som ska avgöra vilken nästa instruktion ska bli, utan blir tvungen att vänta på att detta värde ska beräknas innan nästa instruktion kan börja hämtas. Denna inverkan kan minskas genom att implementera branch prediction.

TSEA28 Datorteknik Y, lösningar till tentamen 161018, reviderad 171020

3. Subrutinen har inga krav på sparande av register eller liknande. Därför används här D1 som mellanresultat och flyttas sist över till D0 innan retur från subrutin. Inga optimeringar är inkluderade (skulle t ex kunna göra xor på 4 byte per gång). Det anges inte om D0 är long, word eller byte, så byte antas. Inget anges om D0 kan vara 0, och vad värdet i så fall ska vara. I lösningen antas D0 är 1 eller större.

a)

```
Subrutin:   move.b      #0,D1       ; nollställ mellanresultat
Loop:       eor.b       (A0)+,D1    ; gör exor mellan packetdata och D1
                                    ; räkna upp adress i A0 med 1
            sub.b       #1,D0       ; minska antal som är kvar att xor:a
            bne         Loop        ; hoppa om inte klar med alla
            move.b      D1,D0       ; flytta xor-värde till D0
            rts                     ; klar, hoppa ur subrutinen
```

b) Mindre felstavning i indata; sista adressen ska vara $0000402C. Resultat av anrop med A0 = $00004028 och D0=$05 blir D0 = $01 xor $51 xor $23 xor $b0 xor $cd = $50 xor $23 xor $b0 xor $cd = $73 xor $b0 xor $cd = $c3 xor $cd = $0e. Alternativt sätt: Skriv upp värden i binär form ovanför varandra. Räkna antal ettor i varje kolumn. Om antal ettor är ojämnt blir värde i den kolumnen 1, annars 0.

4. En avbrottsrutin måste alltid spara alla registervärden innan register används i rutinen. Inget angavs om storleken på utdata sparat i $5000 förutom att maxvärdet är 32. En byte räcker därför som datastorlek.

```
Avbrott:    move.l      D0,-(A7)    ; Spara D0 på stacken
            move.l      D1,-(A7)    ; Spara D1 på stacken
            move.l      $10040,D0   ; Hämta indata (nollställer även avbrottet)
            move.b      #0,D1       ; antal ettor som hittats hittills.
Loop:       lsr.l       #1,D0       ; shift D0 1 bit åt höger, utskiftad bit till C-flaggan
            bcc         zero        ; hoppa om utskiftad bit = 0
            add.b       #1,D1       ; öka antal hittade ettor med 1
Zero:       or.l        #0,D0       ; Kontrollera om någon bit fortfarande är 1 i D0
            bne         Loop
            move.b      D1,$5000    ; Spara resultatet i minnet
            move.l      (A7)+,D1    ; Återställ D1 från stacken
            move.l      (A7)+,D0    ; Återställ D0 från stacken
            rte                     ; Återhopp från avbrott (exception)
```

TSEA28 Datorteknik Y, lösningar till tentamen 161018, reviderad 171020

5. Som titeln på frågan indikerar handlar uppgiften om att förstå hur stacken fungerar. Subrutinen sparar värden på stacken (instruktion på adress $1006) vilket sedan RTS kommer hoppa till. Detta medför att sekvensen av instruktionsadresser blir

Instruktionsadress		D0 efteråt	Stack efter instruktionen utförts
$1000	move.l #$1012,D0	$1012	återhopp
$1006	move.l D0,-(A7)	$1012	$1012, återhopp
$1008	move.l #$102c,D0	$102c	$1012, återhopp
$100e	move.l D0,-(A7)	$102c	$102c, $1012, återhopp
$1010	rts	$102c	$1012, återhopp
$102c	move.l D0,(A7)+	$1012	återhopp
$102e	jsr $100e	$1012	$1032, återhopp
$100e	move.l D0,-(A7)	$1012	$1012, $1032, återhopp
$1010	rts	$1012	$1032, återhopp
$1012	move.l #$101a,D0	$101a	$1032, återhopp
$1018	rts	$101a	återhopp
$1032	rts	$101a	

Slutvärdet i D0 blir alltså $101a och rutinen utför 12 instruktioner innan den är klar.

6. a) 16 Kbyte, 8 byte per cacheline ger 2048 cachelines i minnet totalt.

b) 4-vägs gruppassociativ medför att 512 olika cachelines finns i varje väg. Det behövs då 9 bitar för att välja index i respektive väg. Dessutom behövs 3 bitar för att välja rätt byte i cacheline. Resten blir tag, dvs 32-9-3=20 bitar används för tag.

c) 32-bitars adress, så adressen $1000 är egentligen $00001000 vilket inte påverkar följande beskrivning. Adressen $1000 har tag 1 index 0 ($1000=0001 |0000 0000 0|000 där tag| index|byteposition), , adress $1100 har tag 1 index $20 ($1100=0001 |0001 0000 0|000), adress $1200 har tag 1 index $40 etc. Efter totalt 16 läsningar är nästa adress $2000 som har tag 2 index 0. Gamla värden börjar tas bort när index 0 fås för 5:e gången, dvs 4 läsningar med index 0 har gjorts. Var 16:e läsning är med index 0, så 4 läsningar (då det är 4 vägar i cachen) med index 0 är möjliga innan ett gammalt värde kastas ut. Totalt kan alltså 4*16 = 64 läsningar göras innan gamla värden kastas ut. Svar: 64 läsningar.

TSEA28 Datorteknik Y, lösningar till tentamen 161018, reviderad 171020

7. a) $-7 = -\ 00000111_2 = (11111000+1)_{2C}=11111001_{2C}$.

b) Beräknas som vanlig multiplikation, men MSB är negativ

```
        110011
       *001100
      ---------
+       000000                    + 000000000000
+      000000                     + 00000000000
+     110011       teckenförläng   + 1111110011
+    110011        teckenförläng   + 111110011
+   000000                        + 00000000
-  000000                         - 0000000
  ------------                     --------------
  111101100100                       111101100100
```

Som jämförelse kan det decimala värdet beräknas: $110011_{2C}=-001101_2=-13$, $001100_{2C}=001100_2=12$. Svaret bör därför bli $-13*12=-156 = -(128+16+8+4) = -(0010011100)_2 = (1101100011+1)_{2C} = 111101100100_{2C}$.