TSEA28 Datorteknik Y, lösningar till tentamen 160816

1. a) Antag att hämtfasen redan implementerats, dvs maskininstruktionen har flyttats in i IR-registret.

```
Adress Mikrokod                                Kommentar
  n    buss:=GRx, R:=0, S:=0, AR:=buss,        ; flytta GRa till AR
       uPC:=uPC+1
  n+1  buss:=GRx, R:=0, S:=1, AR=AR+buss       ; Addera GRb till AR
       uPC:=uPC+1                              ; ändra flaggor
  n+2  buss:=AR, R:=0, S:=0, Grx:=buss         ; Spara resultat i GRa, börja
       uPC:=0                                  ; hämtfas för nästa instruktion
```

b) Maskinkoden blir (opcode, GRa, GRb, A) 1101 10 01 00000000 dvs $D900.

c) Utökning till addition med A-fältet: Flytt av IR till AR kommer inkludera opcode och GR och M-fält så dessa oanvända fält måste nollställas innan addition av GRa och GRb görs. Flaggornas värde blir nu beroende på i vilken ordning additionerna görs. I lösningen nedan beräknas GRa := (A+GRa)+ GRb.

```
Adress Mikrokod                                Kommentar
  n    buss:=IR, AR:=buss, uPC:=uPC+1          ; flytta A-fältet till AR
  n+1  buss:=uM (värde $ff), AR:=AR and buss,; maska bort OP,GRx och M bitar
       uPC:=uPC+1
  n+2  buss:=GRx, R:=0, S:=0, AR:=AR+buss,     ; Addera GRa till AR
       uPC:=uPC+1                              ; ändra flaggor
  n+3  buss:=GRx, R:=0, S:=1, AR:=AR+buss,     ; Addera GRb till AR
       uPC:=uPC+1                              ; ändra flaggor
  n+4  buss:=AR, R:=0, S:=0, Grx:=buss         ; Spara resultat i GRa, börja
       uPC:=0                                  ; hämtfas för nästa instruktion
```

2. a) Big endian placerar byte i ordning med mest signifikant byte för i minnet, medan little endian placerar minst signifkant byte först i minnet. Exempel: värdet $12345678 ska sparas i minnet. En big endian maskin placerar $12 på första adressen, medan en little endian placerar $78 på första adressen.

b) Fördel: Protokoll på bussen blir oberoende av processormodell, samma I/O-enheter kan användas på flera datormodeller, överföring över längre avstånd,

Nackdel: Konvertering måste göras mellan aktuell processor och vald buss.

c) Processorns adress (den virtuella adressen) delas upp i två delar, där den mest signifikanta delen används som index i en tabell där start för motsvarande fysiska adress fås. Den minst signfikanta delen av den virtuella adressen läggs sedan till den fysiska startadressen.

d) Halvledarminnen som behåller innehållet vid spänningsbortfall: maskprogrammerade ROM, PROM, EPROM, EEPROM, FLASH. Magnetband etc är inte halvledarminnen.

e) LSR skiftar in en 0:a till vänster, medan ASR behåller värdet på biten längst till vänster (teckenbiten).

3. a) -3 = -(00000011) = 11111100+1=11111101=$FD

b) 5-bitarstalet måste teckenförlängas (anges vara 2-komplementstal) innan additionen sker. $17 = -9, $C2 = -62.

TSEA28 Datorteknik Y, lösningar till tentamen 160816

```
                                    11   11
    10111   teckenförläng            11110111
+11000010                           +11000010
---------                           ---------
                                     10111001
                              =$B9
```

4. En liten miss i uppgiften: Tabell med värde efter bör ha värdet $2E istället för $21 på adresserna $403A och $493C. I lösningen nedan kontrolleras även om anrop görs med längd 0, vilket inte krävs. Om man antar att D0=0 aldrig händer kan subrutinen istället startas från adress Next:.

```
Subrutin:   cmp.l   #0,D0       ; Färdig?
            bne     Next        ; nej, ta nästa varv i loopen
End:        rts                 ; Klar, inga register behöver återställas

Next:       move.b  (A0)+,D1    ; Hämta adress ur listan samt öka A0 med 1
            and.b   #$7F,D1     ; Nollställ bit 7 i tecknet
            cmp.b   #$7F,D1     ; Ska tecken bytas?
            beq     Change      ; Ja, byt
            cmp.b   #$1F        ; Nej, inte $7f, kontrollera istället om $00-$1F
            bgt     Store       ; nej, flytta tecknet
Change:     move.b  #$2E,D1     ; Ersätt med ASCII för '.'
Store:      move.b  D1,(A1)+    ; Spara på rätt plats, öka pekare med 1
            sub.l   #1,D0       ; Kontrollera om fler tecken kvar
            bne     Next:       ; Fortfarande tecken kvar att kopiera/justera
            rts                 ; klar, inga register att återställa
```

5. a) Det finns 8 vägar, så cachen delas upp i 8 likadana parallella delar där varje del är 2^14 byte stor (dvs 16384 byte). Varje sådan del har 8 byte per cacheline, vilket ger 2^11 = 2048 cachelines per väg.

b) Den 32-bitars adress som används delas upp i 3 bitar till val av byte i cacheline, och 11 bitar till val av cacheline i respektive väg. Resten (32-3-11=18) är tag. Varje cacheline behöver därför lagra 18 extra bitar för varje cacheline om 8 byte (8*8=64 bitar). Varje cacheline lagrar därför 18+64=82 bitar. Det finns 8*2^11 cachelines i cachen, så totalt antal bitar blir 82*8*2^11=82*8*2048=82*16384=1343477 bitar.

c) En slumpmässig linjär sekvens av läsningar kan i bästa fall starta på första byte i en cacheline, och i sämsta fall starta på sista byte i en cacheline. Det betyder att i bästa fall kommer hela minnet fyllas (rad efter rad i en väg följt av rad efter rad i nästa väg etc.) tills cacheminnets alla celler fyllts pga läsningarna. När nästa värde sedan läses behöver en tidigare läst cacheline kastas ut. Det betyder att som bäst kan 131072 byte läsas i en sekvens utan att tidigare läst data kastas ut.

I värsta fall startas sekvensen i sista byte av en cacheline, så 7 av de data som läses in aldrig tillhör sekvensen. När 131072-7=131065 byte lästs kommer därför nästa läsning behöva placeras i den 1:a cachelinen som lästes.

Totalt betyder detta att minst 131065 och som mest 131072 byte kan läsas i sekvens innan tidigare läst sekvensdata kastas ut ur cachen.

d) Det finns 8 vägar i cachen, och varje väg behöver jämföra tagvärde från utvald rad med aktuell adress. Det behövs därför 8 jämförare.

6. En liten miss i exemplet; startvärdet på adress $10040 borde vara $28 för att bit 3 (räknat med bit 0 längst till höger) ska bli 1 och tecken kopieras.

```
Avbrott:  move.l  A0,-(A7)      ; Spara undan register som ska användas
          and.b   #$08,$10040   ; Finns tecken att hämta?
          beq     slut          ; Nej, bit 3 var = 0
          cmp.b   #0,$5010      ; finns lediga platser i bufferten?
          beq     slut          ; nej, ingen plats kvar
          move.l  $5000,A0      ; hämta minnesadress för nästa tecken i bufferten
          move.b  $10042,(A0)+; flytta tecken in i buffert, peka på nästa plats
          move.l  A0,$5000      ; Spara position till nästa plats i bufferten
          sub.b   #1,$5010      ; minska platsräknaren med 1
slut:     move.l  (A7)+,A0      ; Återställ A0-register
          rte                   ; avsluta avbrott
```