Tentamen
Datorteknik Y, TSEA28

Datum	2012-06-02
Lokal	TER2, TER4, TERE
Tid	14-18
Kurskod	TSEA28
Provkod	TEN1
Kursnamn	Datorteknik Y
Institution	ISY
Antal frågor	7
Antal sidor (inklusive denna sida)	10
Kursansvarig	Andreas Ehliar
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Andreas Ehliar
Besöker skrivsalen	Around 15 and 17
Kursadministratör	Ylva Jernling
Tillåtna hjälpmedel	Inga
Betygsgränser	**Poäng** **Betyg** 41-50 5 31-40 4 21-30 3 0-20 U

Viktig information

- För de uppgifter där du måste göra uträkningar är det viktigt att du redovisar alla relevanta mellansteg.
- Om du i en viss uppgift ska förklara något, tänk på att du gärna får rita figurer för att göra din förklaring tydligare.
- Uppgifterna i tentan är inte ordnade efter svårighetsgrad.
- Skriv inga svar i detta häfte!

Lycka till!

Fråga 1: Assemblerprogrammering(6p)

På adress $4000 och framåt i det MC68008-baserade tutorsystemet finns det en ASCII-kodad mening enligt specifikationen nedan:

- Ett ord består av bokstäver från A till Z (både stora och små bokstäver).
- Ord separeras av ett eller flera mellanslag.
- Meningen avslutas med en punkt. (Vilket också medför att det sista ordet i meningen även kan avslutas av en punkt.)

Skriv en subrutin i MC68008-assembler som räknar antalet ord i den mening som börjar på adress $4000. Antalet ord ska returneras i register D0. Du kan anta att meningen inte innehåller mer än 2^{14} ord.

Fråga 2: Pipelining(4p)

(a) (2p) Förklara vad pipelining är med maximalt 5 meningar samt en illustration.

(b) (2p) Förklara med maximalt 5 meningar varför villkorliga hopp är problematiska att hantera för en pipelinead dator. Använd gärna illustrationen ifrån föregående deluppgift eller rita en ny illustration.

Fråga 3: DSP-processorer(3p)

En DSP-processor är speciellt anpassad för att kunna utföra vanliga operationer inom signalbehandling på ett effektivt sätt. Nämn tre karaktäristiska egenskaper (utöver pipelining) hos en DSP-processor som gör att den är bättre än generell processor i stil med MC68008 på att snabbt beräkna exempelvis skalärprodukter.

Fråga 4: Mikroprogrammering, flaggor och aritmetik(15p)

I figur 1 återfinns en välkänd mikroprogrammerad dator som används i följande uppgifter:

(a) (7p) Skriv den mikrokod för de instruktioner samt de adresseringslägen som krävs för att följande assemblerprogram ska kunna köras:

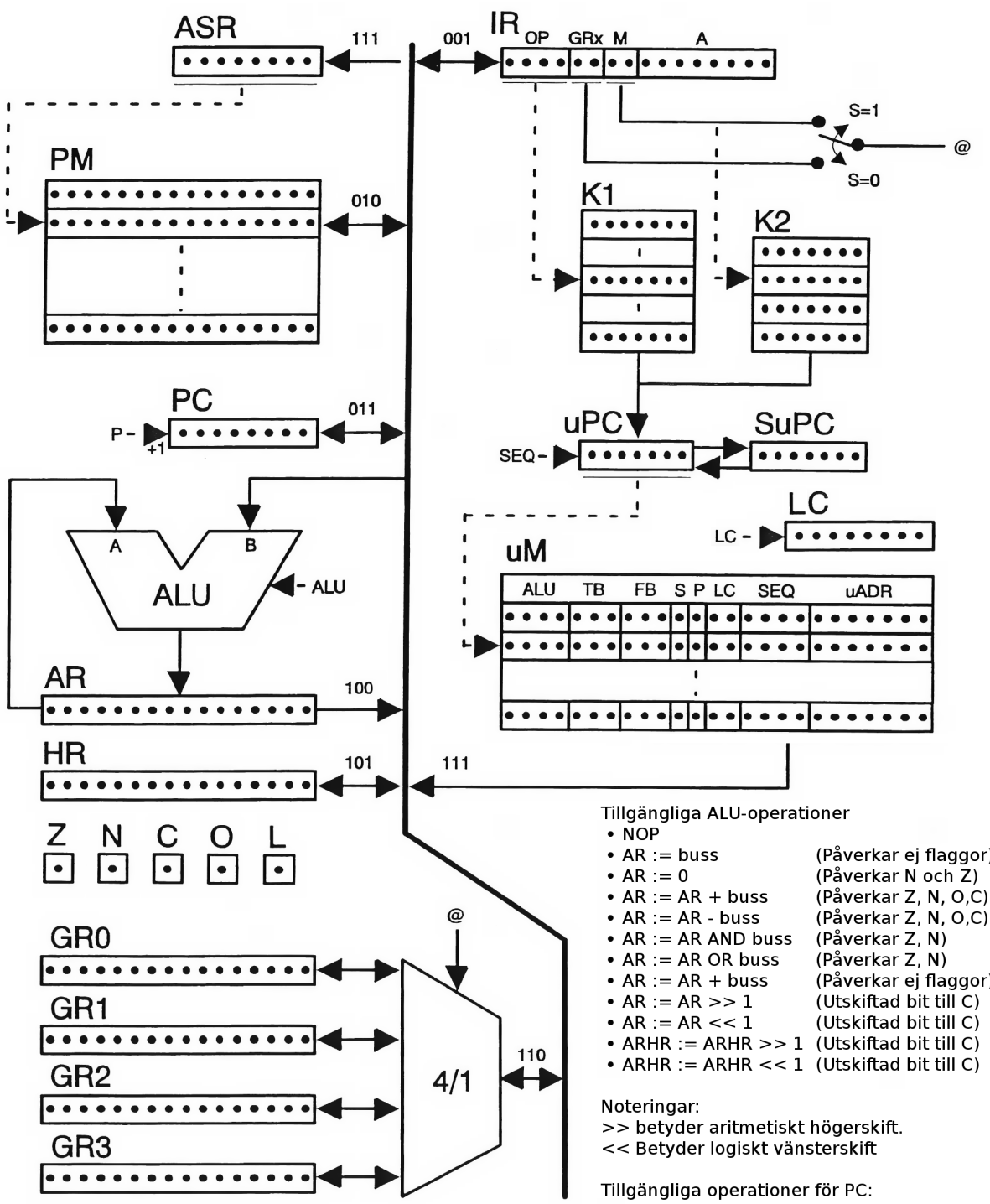
```
rensa:
      PUSH  GR1
      PUSH  GR2
      PUSH  GR3
      LOAD  #0,GR1

loop:
      STORE GR1,(GR3) ; Spara GR1 med indexerad adressering.
      ADD   #1,GR3
      ADD   #1,GR2
      BCC   loop   ; Hoppar om C-flaggan *ej* är satt

      POP   GR3
      POP   GR2
      POP   GR1
      RTS
```

Utöver mikrokoden ska du också specificera vilket innehåll du ska ha i K1 samt K2. GR0 ska användas som stackpekare (och du får anta att M-fältet har värdet 0 i de instruktioner som använder stacken).

(b) (4p) I labdatorn finns det fyra flaggor som har namnen **C**, **Z**, **O** och **N**. (**O**-flaggan kallas för **V** i MC68008.) Förklara vad dessa fyra flaggor används till.

(c) (2p) Förklara tvåkomplementsrepresentation

(d) (2p) Du vill jämföra två stycken tvåkomplementstal för att få reda på vilket tal som är störst. För att göra detta subtraherar du först det ena talet ifrån det andra. Sedan kan du använda flaggorna för att avgöra vilket av de två talen som är störst. Beskriv hur du använder de fyra flaggorna som nämnts ovan för att ta reda på detta!



Tillgängliga ALU-operationer
- NOP
- AR := buss (Påverkar ej flaggor)
- AR := 0 (Påverkar N och Z)
- AR := AR + buss (Påverkar Z, N, O,C)
- AR := AR - buss (Påverkar Z, N, O,C)
- AR := AR AND buss (Påverkar Z, N)
- AR := AR OR buss (Påverkar Z, N)
- AR := AR + buss (Påverkar ej flaggor)
- AR := AR >> 1 (Utskiftad bit till C)
- AR := AR << 1 (Utskiftad bit till C)
- ARHR := ARHR >> 1 (Utskiftad bit till C)
- ARHR := ARHR << 1 (Utskiftad bit till C)

Noteringar:
>> betyder aritmetiskt högerskift.
<< Betyder logiskt vänsterskift

Tillgängliga operationer för PC:
- NOP
- PC := PC + 1 (PC räknas upp med ett)
- PC := buss

Tillgängliga operationer för LC:
- NOP
- LC räknas ned med ett
- LC laddas från uADR

Tillgängliga operationer för uPC:
- uPC := uPC + 1 (uPC räknas upp med ett)
- uPC := K1(OP)
- uPC := K2(M)
- uPC := 0
- uPC := uADR
- uPC := uADR om C är 1, annars uPC+1
- uPC := uADR om L är 1, annars uPC+1

Under varje klockcykel kan även en av följande enheter skicka data från bussen: IR, PM, PC, AR, HR, GRx eller uM. En av följande enheter kan också ta emot data ifrån bussen varje klockcykel: IR, PM, PC, AR, HR, GRx eller ASR Viktigt: När uM (de 16 minst signifikanta bitarna) skickas ut till bussen kan enbart en ALU-operation och/eller en bussöverföring göras. uPC räknas också automatiskt upp med ett i detta fall.

Slutligen så används signalen S för att välja om M eller IR-fältet ska användas till att adressera GR0-GR3.

Figur 1: En välkänd mikroprogrammerad dator. (Björn Lindskog 1981)

Fråga 5: Avbrott samt förståelse av assemblerprogram(9p)

Till det tutorsystem du har använt i laborationerna har du kopplat in en tryckknapp på pinne noll på PIAB (adress $10082). På avbrottsingången på PIAB har du kopplat in en klocka som skickar ut en fyrkantsvåg med frekvensen 1 Hz.

Du har också skrivit in följande program:

```
huvudprogram:
    move.l #$7000,a7
    jsr  initiera_parallelport
    jsr  initiera_avbrott
    move.w #0,$5000

ingenknapp:
    move.b $10082,d0
    and.b  #1,d0
    beq    ingenknapp

    * BUG1
    move.b $5000,d0
    move.b $5001,d1
    * BUG2
    add.b  #48,d1
    jsr    printchar

    move.b d0,d1
    add.b  #48,d1
    jsr    printchar

    move.b #10,d1
    jsr    printchar

    bra    ingenknapp
```

```
avbrott_1hz:
    * BUG3
    tst.b  $10080

    move.b $5000,d0
    cmp.b  #9,d0
    beq    nolla_ental

    add.b  #1,d0
    move.b d0,$5000

    bra    avbrott_klart

nolla_ental:

    move.b #0,$5000
    move.b $5001,d0

    cmp.b  #5,d0
    beq    nolla_tiotal

    add.b  #1,d0
    move.b d0,$5001
    bra    avbrott_klart

nolla_tiotal:
    move.b #0,$5001

avbrott_klart:
    * BUG4
    rte
```

- `initiera_parallelport` sätter upp parallelporten så att tryckknappen är en ingång
- `initiera_avbrott` sätter upp parallelporten så att avbrottet på PIAA aktiveras när en stigande puls inkommer. Den sätter också upp motsvarande avbrottsvektorer så att den pekar på `avbrott_1hz`. Slutligen sänker den avbrottsnivån.
- `printchar` skriver ut det ASCII-kodade tecken som ligger i d1 i terminalfönstret. (Den fungerar på samma sätt som `printchar` gjorde i lab 1.)

(a) (4p) Vad kommer detta program att göra?

(b) (4p) Olyckligtvis har detta program några allvarliga fel. På raderna markerad med `BUG1-BUG4` saknas följande instruktioner (som är listade utan inbördes ordning):

- `move.l (a7)+,d0`
- `and.w #$f8ff,sr`
- `or.w #$0700,sr`
- `move.l d0,-(a7)`

Matcha ihop dessa instruktioner med rätt `BUG*` samt **förklara** varför programmet inte kommer att fungera korrekt om dessa rader saknas.

(c) (1p) Vad är skillnaden mellan instruktionerna `RTS` och `RTE`? (Det räcker inte att säga att den ena används i avbrottshanterare och att den andra används i subrutiner.)

Fråga 6: Minnen och bussar(8p)

(a) (2p) Förklara vad konceptet buss innebär i datortekniksammanhang.

(b) (3p) Förklara kortfattat hur en cache fungerar.

(c) (2p) Förklara konceptet cacheline och associativitet

(d) (1p) Varför behövs en cache i en modern persondator?

Fråga 7: A/D-omvandling(5p)

(a) (3p) Beskriv kortfattat hur A/D-omvandling med hjälp av successiv approximation, reversibel räknare samt flash-omvandling fungerar. (Tips: Om du ritar figurer här behöver du bara skriva 1-2 meningar per omvandlare.)

(b) (2p) Diskutera hur dessa A/D-omvandlare förhåller sig till varandra gällande samplingsfrekvens samt storlek.

Utdrag ur en ASCII-tabell

Kodning	Tecken	Kodning	Tecken	Kodning	Tecken	Kodning	Tecken
10	Nyrad	32	Mellanslag	46	Punkt		
48	0	67	C	86	V	105	i
49	1	68	D	87	W	106	j
50	2	69	E	88	X	107	k
51	3	70	F	89	Y	108	l
52	4	71	G	90	Z	109	m
53	5	72	H	91	[	110	n
54	6	73	I	92	\	111	o
55	7	74	J	93	]	112	p
56	8	75	K	94	^	113	q
57	9	76	L	95	_	114	r
58	:	77	M	96	‘	115	s
59	;	78	N	97	a	116	t
60	<	79	O	98	b	117	u
61	=	80	P	99	c	118	v
62	>	81	Q	100	d	119	w
63	?	82	R	101	e	120	x
64	@	83	S	102	f	121	y
65	A	84	T	103	g	122	z
66	B	85	U	104	h		

Kort repetition av M68000

Mnemonic	Kort förklaring	Mnemonic	Kort förklaring
ADD	Addition	EXG	Exchange
ADDX	Add with X flag	EXT	Sign extend
AND	Logic and	EXTB	Sign extend a byte to 32 bit
ASL	Arithmetic shift left	LEA	Load effective address
ASR	Arithmetic shift right	LSL	Logic shift left
BCC	Branch on carry clear	LSR	Logic shift right
BCS	Branch on carry set	MOVE	Move
BEQ	Branch on equal	MULS	Signed multiplication
BGT	Branch on greater than	MULU	Unsigned multiplication
BLT	Branch on less than	NEG	Negate
BNE	Branch on not equal	NOP	No operation
BRA	Branch always	OR	Logic OR
BSR	Branch to subroutine	ROL	Rotate left
CLR	Clear	ROR	Rotate right
CMP	Compare (Destination - Source)	RTE	Return from exception
DIVS	Signed division	RTS	Return from subroutine
DIVU	Unsigned division	SUB	Subtract
EOR	Logic XOR	TST	Set integer condition codes

```
; Exempel på M68000 kod som kopierar 200 bytes från $2000 till $3000
    LEA $2000,A0
    LEA $3000,A1
    MOVE.B #50,D0
loop
    MOVE.L (A0)+,(A1)+
    ADD.B #-1,D0
    BNE loop
```

Errata

Några smärre fel hade smugit sig in i tentan som har rättats till denna version som ligger på hemsidan:

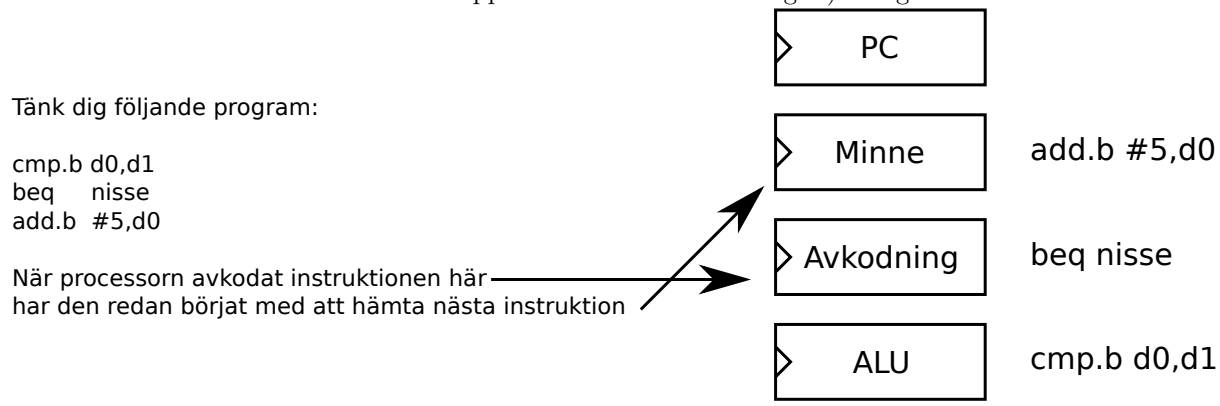
- I 5a har #10082 rättats till $10082. Och move.b d1,$5001 har rättats till move.b d0,$5001
- I 5b har and.b och or.b rättats till and.w och or.w
- I ASCII-tabellen har ASCII-koden för punkt lagts till.

Lösningsförslag uppgift 1

```
      move.l #$4000,a0
      clr.w  d0

loop_noword: move.b (a0)+,d1
      cmp.b  #'.',d1
      bne    notdone_noword  ; Slut på meningen?
      rts

notdone_noword:
      cmp.b #' ',d1     ; Vänta på att ett ord börjar.
      beq loop_noword

      add.w #1,d0       ; Öka på ordräknaren

loop_word: move.b (a0)+,d1
      cmp.b #'.',d1     ; Slut på meningen?
      bne notdone_word
      rts

notdone_word:
      cmp.b #' ',d1
      bne loop_word
      bra loop_noword
```

Lösningsförslag uppgift 2

a) Se läroboken/Daniels kompendium.
b) När ett villkorligt hopp förekommer i en pipelinead dator kommer datorn att läsa in några instruktioner innan datorn känner till om hoppvillkoret är sant eller falskt. (Eller för den delen innan datorn känner till huruvida instruktionen är en hoppinstruktion överhuvudtaget.) Se figur nedan:



Lösningsförslag uppgift 3

Exempel på karaktäristiska egenskaper:

- Specialla MAC enheter som med hjälp av enbart en instruktion kan både multiplicera och addera vilket ofta behövs i signalbehandlingsalgoritmer såsom faltning och skalärprodukter.
- Minnessystemet är anpassat för att kunna leverera två operander samtidigt till ovan nämnda MAC-enhet genom att det finns minst tre minnen (två för data och ett för programmet).
- Genom att ha speciella loop-instruktioner så undviker man onödiga villkorliga hoppinstruktioner i de kritiska inre looparna.

Lösningsförslag uppgift 4

a) Se mikroprogrammeringslaborationerna. (Förutom BCC som inte torde innebära några större svårigheter jämfört med BNE.)
b)

- C-flaggan sätts om svaret för en operation hamnat utanför talområdet för teckenlösa tal vid addition samt om en etta skiftats ut vid skiftoperationer
- Z-flaggan sätts om svaret för en operation är noll
- O-flaggan (V-flaggan) visar om svaret för en operation hamnat utanför talområdet när tvåkomplementsrepresentation används
- N-flaggan visar om svaret för en operation är negativ om tvåkomplementsrepresentation används

c) I ett tvåkomplementsrepresenterat tal har den mest signifikanta siffran en negativ vikt istället för en positiv vikt. Det vill säga, i ett N bitars tal så har bitarna följande vikter räknat från vänster till höger: -2^{N-1}, 2^{N-2}, 2^{N-3}, ..., 2^1, 2^0.

(Den största fördelen med detta är att samma adderare/subtraherare kan användas för både teckenlösa tal och tal med tecken.)
d) Vid operationen A - B sätts flaggorna på ett sådant sätt så att uttrycket $N \; xor \; O$ är sant om B är störst. (Se även sorteringsuppgiften i mikroprogrammeringslaborationen.)

Lösningsförslag uppgift 5

Tyvärr råkade fel version av detta program smyga sig in i tentan. Lyckligtvis så är felen tämligen uppenbara, se erratan ovan. Information om de buggar av det mer allvarliga slaget som fanns med i tentan har givits muntligt på tentatillfället till alla som satt kvar efter att den första tentanden upptäckte dessa buggar.

a) Programmet hanterar två värden som finns på position $5000 samt $5001 i minnet. Värdet på position $5001 tolkas som en tiotalssiffra och $5000 som ental.

Huvudprogrammet väntar först på en knappnedtryckning. Sedan ASCII-kodar den dessa värden och skriver ut dessa på terminalen, följt av en nyrad för att sedan återgå till att vänta på en knapptryckning.

Avbrottsrutinen räknar varje sekund upp värdet ifrån 00 till 59 för att sedan återgå till 00 igen.

Det vill säga, när du trycker på knappen får du reda på hur många sekunder som gått sedan programmet startats, modulo 60, eller med andra ord, programmet hanterare sekunderna i en digital klocka.
b)

- BUG1 - or.b #$0700,sr
- BUG2 - and.b #$f8ff,sr
- BUG3 - move.l d0,-(a7)
- BUG4 - move.l (a7)+,d0

Utan att vi höjer avbrottsnivån och sedan sänker den med or.b/and.b kan det hända att avbrottsrutinen körs mellan det att $5000 läses och att $5001 läses i huvudprogrammet. Om avbrottsrutinen räknar upp både $5000 samt $5001 kan det innebära att ett helt felaktigt värde matas ut på serieporten. För att förhindra detta måste vi alltså stänga av interrupt i denna kritiska sektion.

Vad gäller BUG3/BUG4 så måste vi givetvis spara undan d0 eftersom avbrottsrutinen annars kommer att förstöra innehållet i detta register. (Om man vill vara riktigt pedantiskt så kan man också påpeka att det inte finns några indikationer i programmet på att d0 har reserverats enbart för avbrottshanteraren.)

c) Här räcker det med att säga att RTE återställer både statusregistret och programräknaren ifrån stacken medans RTS enbart återställer programräknaren.

Lösningsförslag uppgift 6

a) Se kurslitteraturen. För full poäng vill jag här att man tar upp något mer än enbart att det är ett sätt att ansluta enheter till varandra.
b) Se kurslitteraturen. För full poäng vill jag exempelvis att det ska framgå att cachen fungerar utan att programmeraren behöver vara involverad på något sätt, det vill säga att cachen automatiskt kollar vilken adress som processorn vill åt. (Det är många som har skrivit en förklaring i stil med att cachen är ett snabbt SRAM som sitter nära processorn.) Detta är dock inte tillräckligt för mer än en poäng eftersom man kan sätta ett snabbt SRAM nära CPU:n utan att det för den skull är en cache. (Detta är för övrigt hur SPE-kärnorna i PlayStation 3 fungerar.)
c) Se kurslitteraturen.
d) Här vill jag att det ska framgå att externt DRAM är mycket långsammare än internt SRAM. Ett antal personer har fått full poäng på denna fråga genom att detta framgår i svaret på deluppgift b trots att svaret på fråga d inte är tillfredsställande. (Ett fåtal personer har dock fått fulla poäng här genom att beskriva någonting annat här som tyder på att de har god kunskap om hur en cache fungerar och varför den behövs i en modern dator.)

Lösningsförslag uppgift 7

a) Se kurslitteraturen. Full poäng nås lättast genom att rita en figur för A/D-omvandlaren som är implementerad med hjälp av en reversibel räknare som visar både den analoga insignalen samt hur utsignalen ifrån D/A-omvandlaren approximerar insignalen. Samma sak för A/D-omvandlaren med successiv approximation. För flash-omvandlaren räcker det med en figur som visar på hur den är implementerad. (Till alla figurer måste åtminstone lite förklarande text finnas.)
b) Se kurslitteraturen. Tyvärr är det många som har missat den här uppgiften genom att de enbart har diskuterat hastigheten hos en A/D-omvandlare implementerad med hjälp av reversibel räknare och successiv approximation och inte diskuterat storleken. (Det minsta som krävs för poäng är antingen att hastigheten för alla tre typer diskuteras eller att storlek samt hastighet diskuteras för två typer.)

Notering: Det är förvånande att det är såpass många som inte kan diskutera hastigheten hos en flash-omvandlare eftersom en A/D-omvandlare av den typen användes i A/D-laborationen.