# Tentamen

## Datorteknik Y, TSEA28

| | |
|---|---|
| *Datum* | 2024-01-05 |
| *Lokal* | TER2, TER4 |
| *Tid* | 14-18 |
| *Utb. kod* | TSEA28 |
| *Modul* | TEN1 |
| *Kursnamn*<br>*Provnamn* | Datorteknik Y<br>Skriftlig tentamen |
| *Institution* | ISY |
| *Antal frågor* | 6 |
| *Antal sidor (inklusive denna sida)* | 6 |
| *Kursansvarig* | Kent Palmkvist, 1347, Kent.Palmkvist@liu.se |
| *Lärare som besöker skrivsalen*<br>*Telefon under skrivtiden* | Kent Palmkvist<br>013-281347 |
| *Besöker skrivsalen* | Ca kl 15 och kl 17 (+/- 30 minuter) |
| *Kursadministratör* | Ulrika Ericsson, 013 282379 |
| *Tillåtna hjälpmedel* | Inga |
| *Betygsgränser* | Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5 |
| *Visning* | Fredag 19/1 kl 14-15 i examinators kontor |

## Viktig information

- Hexadecimala värden anges antingen som 0x1234 eller $1234
- För de uppgifter där du måste göra uträkningar är det viktigt att du redovisar alla relevanta mellansteg
- I de uppgifter där du ska skriva assembler- eller mikrokod är det viktigt att du kommenterar koden
- 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: Mikroprogrammering (10p)

I figur 1 återfinns en bekant datormodell som du ska mikroprogrammera i denna uppgift.

Jämfört med den modell du sett tidigare har följande förändring gjorts:
Mikroprogrammerings-ordet har utökats med ytterligare en bit (kallad R). Om R=0 fungerar datorn precis som tidigare. Om R=1 sätts styrsignaler som styr multiplexern vid GR0-GR3 till 3 (dvs register GR3 väljs), oavsett vad IR innehåller.

Notera att det är viktigt att du skriver vilken additionsoperation du valt (den som påverkar flaggorna eller den som inte påverkar flaggorna). (Om du inte skriver något speciellt kommer jag att anta att du ej vill modifiera flaggorna).

Mikrokodsminnnet innehåller i denna datormodell bland annat

| addr | Mikrokod | Kommentar |
|---|---|---|
| 0 | ASR := PC, uPC := uPC+1 | |
| 1 | IR := PM, PC := PC+1, uPC := uPC+1 | |
| 2 | uPC := K2(M) | |
| 3 | ASR := IR, uPC := K1(OP) | ; direktadressering (M=00) |
| 4 | ASR := PC, uPC := uPC + 1 | ; omedelbar adressering (M=01) |
| 5 | PC := PC + 1, uPC := K1(OP) | |
| 6 | ASR := IR, uPC := uPC+1 | ; indirekt adressering (M=10) |
| 7 | ASR := PM, uPC := K1(OP) | |
| 8 | AR := IR, uPC := uPC+1 | ; indexerad adressering (M=11) |
| 9 | AR := GR3+AR, uPC := uPC+1, R:=1 | |
| 0xa | ASR := AR, uPC := K1(OP) | |

(a) (6) Skriv mikrokod för instruktionen POP GRx. Instruktionen ska hämta ett värde från stacken och uppdatera stackpekaren (GR3 används som stackpekare). GR3 pekar på första lediga plats på stacken, och stacken växer mot lägre adresser. Antag att adresseringsmode alltid är 00, och A-fältet alltid är satt till 0. Flaggor får påverkas. Ange adresserna för mikrokodsraderna som absoluta numeriska värden.

**Exempel**: Antag GR3=0x0034, PM(0x35)=0x1234. Om instruktionen POP GR2 utförs ska resultera i GR2=0x1234 och GR3=0x0035.

(b) (2) Antag POP-instruktionen har opcode $1101_2$. Ange maskininstruktionen för POP GR2 i hexadecimal form.

(c) (2) Ange innehåll i K2. Ange alla värden (både för adress och data) i binär form.

## Fråga 2: Allmän teori (10p)

(a) (2) Ange två skillnader mellan SRAM och DRAM

(b) (2) Vad skiljer en SIMD-dator från en vanlig generell dator?

(c) (2) Varför innehåller en dator alltid någon form av ROM-minne?

(d) (2) Ge ett litet kodexempel på hur man kan programmera en processor med delayed branch så styrkonflikten inte sänker prestandan hos processorn.

(e) (2) Ange två egenskaper hos ett fleranvändarsystem som kräver att en MMU ingår i datorn.

## Fråga 3: Assemblerprogrammering (10p)

Skriv en subrutin som skriver ut en ASCII sträng byte för byte med speciell hantering av styrtecken. Subrutinen ska för ASCII-värden mellan 0x20 och 0x7E skriva ut dom direkt. För ASCII-värden mellan 0xa0 och 0xfe ska först bit 7 nollställas i värdet och sedan det nya värdet skrivas ut direkt. För alla andra värden ska dom istället skrivas ut som hexadecimala värden, där först två tecken ’0’ (ASCII-kod 0x30) och ’x’ (ASCII-kod 0x78) skrivs ut följt av mest signficant nibble och sedan minst signifikant nibble i hexadecimal form.

ASCII-värdet för ’0’ är 0x30, ’1’ är 0x31, ... ’9’ är 0x39, ’A’ är 0x41, ’B’ är 0x42, ..., ’F’ är 0x46.

För utskrift av ett tecken ska en redan skriven subrutin kallad Subrutin1 användas, som förväntar sig att r0 innehåller ASCII-värdet för det tecken som ska skrivas ut. Subrutinen Subrutin1 förstör även värdet i register r1 och r2. Subrutin1 ska inte skrivas.

De byte som ska skrivas ut finns lagrad i minnet med start på den adress som anges i r0. Antal byte som ska skrivas ut anges i r1. Minnescellerna som innehåller strängen får inte ändras av subrutinen.

**Exempel**: r0 = 0x20001005, r1=0x00000006

Minnesinnehåll

| Adress | Värde | | | |
|---|---|---|---|---|
| 0x20001000 | 0x10 | 0x76 | 0x86 | 0x9A |
| 0x20001004 | 0x21 | 0x48 | 0x69 | 0x4A |
| 0x20001008 | 0xCF | 0x1E | 0x6F | 0x70 |
| 0x2000100C | 0x70 | 0x21 | 0x00 | 0x02 |

Värdet 0xCF får då bit 7 = 0 och blir 0x4F, och värdet 0x1E skrivs ut som de fyra tecknen ’0’ ’x’ ’1’ ’E’. Subrutinen Subrutin1 anropas därför 9 gånger med r0 satt följande värden: 0x48, 0x69, 0x4A, 0x4F, 0x30, 0x78, 0x31, 0x45, 0x6F.

## Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som först läser ett 8-bitars 2-komplementvärde från adress 0x40001000 och teckenförlänger detta till 16 bitars längd. Därefter ska ett 16 bitars 2-komplementvärde läsas från adress 0x40001010. De två värdena ska sedan adderas ihop och denna summa ska placeras i r0 (med bit 31 till bit 16 = 0). Därefter ska subrutinen Subrutin2 anropas. Subrutin2 ska inte skrivas. Subrutin2 förstör r0, r4 och r5.

Efter att subrutin2 är klar ska det största av de två 16-bitarsvärdena skrivas till minnet på adress 0x40001020.

Inga andra avbrott får starta medan denna avbrottsrutin kör.

**Exempel**: Avbrottsrutinen läser in värdet 0x87 från adress 0x40001000. Teckenförlängt blir detta värde 0xff87. Därefter läses värdet 0x4413 in från minnesadress 0x40001010. Summan blir 0x439A (nollställer bit 31 till bit 16) som läggs i r0. Subrutin2 anropas. Slutligen skrivs värdet 0x4413 till adress 0x40001020 eftersom i 16-bit tvåkomplementrepresentation är 0x4413 > 0xff87.

## Fråga 5: Aritmetik (6p)

(a) (2p) Antag register r0=0x12345678 och r1=0x6543210 och instruktionen AND r0,r1 körs. Vad blir värdet i r0 och värdet på flaggorna Z och N?

(b) (2p) Beräkna värdet i decimal form för 2-komplementsvärdet $10101_{2C}$.

(c) (2p) Beräkna summan av 6-bitars 2-komplementstalet $011101_{2C}$ med 4-bitars 2-komplementstalet $1101_{2C}$. Ange summan som ett 10-bitars 2-komplementstal.

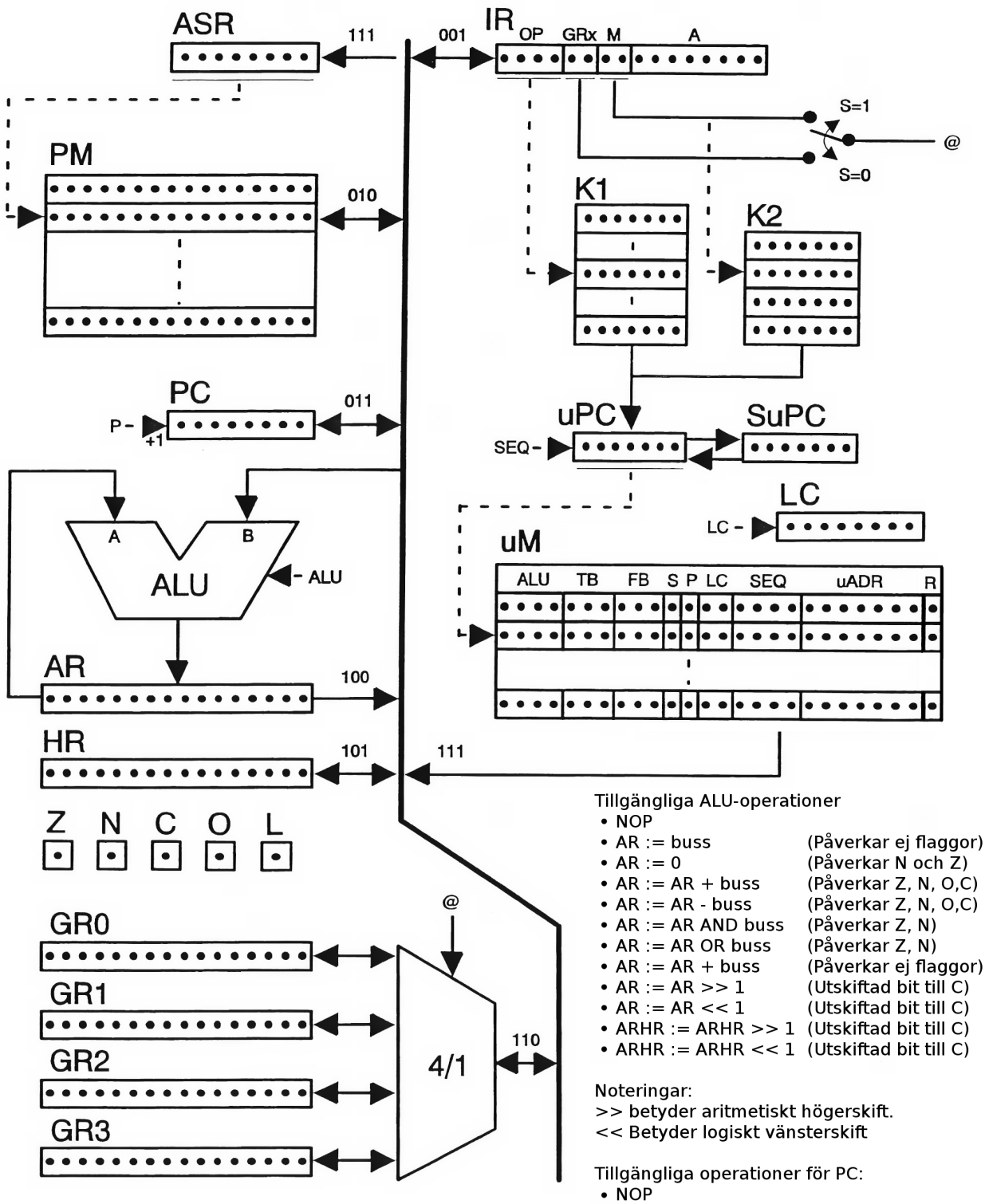
## Fråga 6: Cache (6p)

En processor har en gruppassociativ cache på 16KB (dvs 16384 byte). Adressbussen är 32 bitar lång. Varje cacheline är 64 byte lång. 20 adressbitar används som tag av cachen.

(a) (2p) Vilken associativitet har denna cache?

(b) (2p) Hur många adressbitar används som index?

(c) (2p) Antag cachen är tom. Hur många cachemissar fås om 8 läsninger enligt sekvensen 0x2456789E + n*0x006 (n = 0,1,2,3,..7.) görs? Antag att varje läsning hämtar ett 16-bitars tal.




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 (valfri) flagga är 1, annars uPC+1
- uPC := uADR om (valfri) flagga är 0, 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.

Signalen S väljer om M eller GRx-fältet ska användas till att adressera GR0-GR3 (S=0 innebär att GRx-fältet adresserar GR0-GR3). Slutligen används signalen R=1 för att tvinga styrsignalen @ att bli 3.

*Figur 1: En välkänd datormodell (Björn Lindskog 1981)*

Kort repetition av ARM Cortex-M4

| Mnemonic | Kort förklaring | Mnemonic | Kort förklaring |
|---|---|---|---|
| ADR | Address | CPSID | Disable interrupt |
| ADD, ADDS | Addition | CPSIE | Enable interrupt |
| ADDC,ADDCS | Addition with C flag | EOR,EORS | Logic XOR |
| AND,ANDS | Logic AND | LDR | Load register |
| ASR,ASRS | Arithmetic shift right | LDRB,LDRSB | Load register byte |
| B | Branch | LDRH,LDRSH | Load register halfword |
| BCC,BLO | Branch on carry clear | LSL,LSLS | Logic shift left |
| BCS,BHI | Branch on carry set | LSR,LSRS | Logic shift right |
| BEQ | Branch on equal | MOV,MOVS | Move |
| BGE | Branch on greater than or equal | MUL,MULS | Multiply |
| BGT | Branch on greater than | MVN | Move not |
| BL | Branch and link | NOP | No operation |
| BLT | Branch on less than | ORR,ORRS | Logic OR |
| BLE | Branch on less than or equal | POP | Pop |
| BLS | Branch on lower or same | PUSH | Push |
| BLT | Branch on less than | ROR | Rotate right |
| BMI | Branch on minus | SBC,SBCS | Subtraction with C flag |
| BNE | Branch on not equal | STR | Store register |
| BPL | Branch on plus | STRB | Store register byte |
| BVC | Branch on overflow clear | STRH | Store register halfword |
| BVS | Branch on overflow set | SUB,SUBS | Subtraction |
| BX | Branch and exchange | TST | Test |
| CMP | Compare (Destination - Source) | | |

```
; Exempel på ARM Cortex-M4 kod som kopierar 200 bytes från 0x2000 till 0x3000

main:  mov  r0,#(0x2000 & 0xffff)
       movt r0,#(0x2000 >> 16)
       mov  r1,#(0x3000 & 0xffff)
       movt r1,#(0x3000 >> 16)
       mov  r3,#50
loop:
       ldr  r4,[r0],#4
       str  r4,[r1],#4
       subs r3,r3,#1
       bne  loop
```