# Tentamen

## Datorteknik Y, TSEA28

| | |
|---|---|
| *Datum* | 2021-08-18 |
| *Lokal* | TER1, FE245(F) |
| *Tid* | 8-12 |
| *Kurskod* | TSEA28 |
| *Provkod* | 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>1347, 013-281347 |
| *Besöker skrivsalen* | Ca kl 9 och kl 11 (+/- 30 minuter) |
| *Kursadministratör* | |
| *Tillåtna hjälpmedel* | Inga |
| *Betygsgränser* | Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5 |
| *Visning* | Fredag 3 September kl 13.00 – 14.00 i kursansvarigs 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 ni sett tidigare har följande förändring gjorts: Mikroprogrammeringsordet 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 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,  PC := PC+1, uPC := K1(OP) | ; omedelbar adressering (M=01) |
| 5 | ASR := IR, uPC := uPC+1 | ; indirekt adressering (M=10) |
| 6 | ASR := PM, uPC := K1(OP) | |
| 7 | AR := IR, uPC := uPC+1 | ; indexerad adressering (M=11) |
| 8 | AR := GR3+AR, uPC := uPC+1, R:=1 | |
| 9 | ASR := AR, uPC := K1(OP) | |

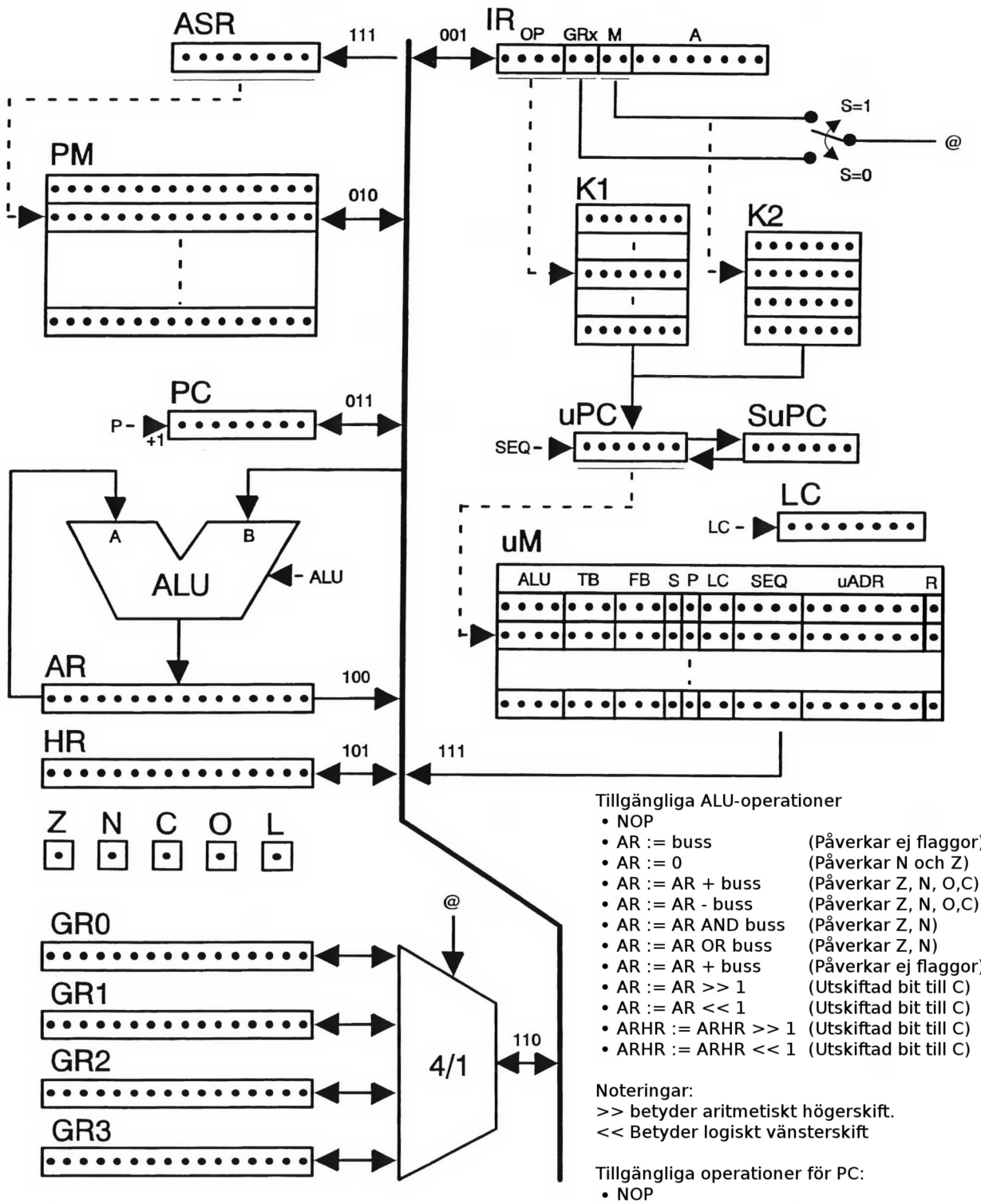
(10p) Skriv mikrokod för instruktionen SWAPLT GRx,M,A. Instruktionen ska testa om flaggorna indikerar att en tidigare jämförelse/subtraktion av tvåkomplementtal gett ett negativt svar, och om det var ett negativt resultat ska värdet i register GRx byta plats med värdet som anges av adresseringsmode och A-fält. Notera att SWAPLT inte ska utföra något egen subtraktion/jämförelse.

Adresseringsmoderna 00, 10 och 11 ska stödjas (antag att adresseringsmode 01 aldrig kan hända). Flaggorna ska inte påverkas av instruktionen.

Ange även adresserna för mikrokodsinstruktionerna som absoluta tal.

**Exempel1:** GR1 = 0x0123, PM(0x20)=0x4321, senaste instruktion som påverkade flaggorna beräknade 0x1230 – 0x5667. När instruktionen SWAPLT GR1, 00,0x20 körs ska resultatet bli GR1 = 0x4321 och PM(0x20)=0x0123.

**Exempel2:** GR3 = 0x5678, PM(0x33)=0x9809, senaste instruktion som påverkade flaggorna beräknade 0xfa00 – 0x9102. När instruktionen SWAPLT GR3, 00,0x33 körs ska varken minne eller GR1 påverkas.




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)*

## Fråga 2: Allmän teori (10p)

(a) (2p) Ange två nackdelar med att ha dedikerade I/O instruktioner jämfört med att ha minnesmappad I/O.

(b) (2) Beskriv skillnad i mängd hårdvara (minne och logik) för en 16 Kbyte 2-vägs gruppassociativ cache jämfört med mängd hårdvara i en 16 Kbyte fullt associativ cache.

(c) (2) Kan instruktionen bx lr ge upphov till en styrkonflikt i en pipelinad ARM-processor? Motivera ditt svar.

(d) (2) Beskriv skillnaden mellan von Neumann och Hardward arkitektur.

(e) (2) Vad står förkortningen EPROM för, och vilka egenskaper (volatilitet, skrivbarhet) har detta minne?

## Fråga 3: Assemblerprogrammering (10p)

I minnet ligger ett antal 32-bitars 2-komplementstal lagrade. Direkt efter varje tal ligger ett 32-bitars värde som är adressen till nästa tal. Adressen efter sista talet i minnet har värdet 0. Talen är alltid placerade på en adress som är jämnt delbar med 4.

Skriv en subrutin som anropas med adressen till första talet placerat i register r0. Subrutinen ska sedan summera alla talen som 64-bitars tal och returnera med summan placerad i register r0 och r1 där r0 har minst signfikant del (lägsta 32 bitarna i summan) och r1 har mest signifikant del (högsta 32 bitarna i summan).

**Exempel**: subrutinanrop med r0 = 0x20001004

| Adress | Värde | Adress | Värde |
|---|---|---|---|
| 0x20001000 | 0x12345678 | 0x20001018 | 0xabcdef01 |
| 0x20001004 | 0x10203040 | 0x2000101c | 023456789 |
| 0x20001008 | 0x20001020 | 0x20001020 | 0x80101020 |
| 0x2000100c | 0x01010205 | 0x20001024 | 0x20001010 |
| 0x20001010 | 0x90010203 | 0x20001028 | 0xa0020020 |
| 0x20001014 | 0x20001028 | 0x2000102c | 0x00000000 |

Subrutinen beräknar i exemplet summan av 0x10203040 + 0x80101020 + 0x90010203 + 0xa0020020. Eftersom en 64-bitars summa ska beräknas behöver negativa tal teckenförlängas först. Summan blir därför 0x0000000010203040 + 0xffffffff80101020 + 0xffffffff90010203 + 0xffffffffa0020020 = 0xfffffffec0334283. Subrutinen returnerar därför i detta exempel r0=0xc0334283 och r1=0xfffffffe.

## Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som ska läsa fyra 8-bitars värden (positiva heltal) från I/O-porten 0x40001000, summera dessa värden, och slutligen skriva tillbaks värdena i omvänd ordning till I/O-porten och spara summan i minnet på adress 0x20001000 som ett 16-bitars tal.

Bara 1 byte får läsas från porten per läsning. Bara 1 byte får skrivas till porten per skrivning.

**Exempel**: Avbrottet läser först värdet 0x80 från adress 0x40001000, därefter vädet 0x9a från adress 0x40001000, därefter värdet 0x23 från adress 0x40001000 och slutligen värdet 0x56 från adress 0x40001000. Därefter skrivs värdet 0x56 till adress 0x40001000, därefter värdet 0x23 till adress 0x40001000, därefter värdet 0x9a till adress 0x40001000, och slutligen värdet 0x80 till adress 0x40001000. Summan 0x80+0x9a+0x23+0x56 = 0x193 skrivs som ett 16-bitars värdet till adress 0x20001000.

## Fråga 5: Aritmetik (6p)

(a) (2p) Översätt det 7 bitars 2-komplementstalet 1001101 till decimal form

(b) (2p) Antag en 8-bitars dator beräknar summan av de hexadecimala värdena 0x83 och 0xd9. Ange summan i binär form och värdet på flaggorna Z, N, C och O.

(c) (2p) Ställ upp och beräkna den binära addition 42 + (-5). Svaret ska representeras med 8 bitar.

## Fråga 6: Cache (6p)

Apples M1 ARM processor har en instruktionscache som är 128 Kbyte stor. Addressbussen är 64 bitar stor. Antag att varje cache line är 128 bitar lång och att cachen är 8-vägs gruppassociativ.

(a) (2p) Hur många bitar av adressen används som tag?

(b) (2p) Antag cachen är tom. Antag 5 läsningar görs i sekvensen 0x19C + n*6 (n=0,1,2,3,4). Hur många cachemissar fås?

(c) (2p) Antag cachen är tom. Hur många läsningar i sekvensen 0x12345678 + n*400 (n=0,1,2,3,....). kan göras innan redan inläst data börjar kastas ut ur cachen?

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 negative |
| 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
```