

Tentamen

Datorteknik Y, TSEA28

<i>Datum</i>	2021-10-20
<i>Lokal</i>	TER1
<i>Tid</i>	8-12
<i>Kurskod</i>	TSEA28
<i>Provkod</i>	TEN1
<i>Kursnamn</i> <i>Provnamn</i>	Datorteknik Y Skriftlig tentamen
<i>Institution</i>	ISY
<i>Antal frågor</i>	6
<i>Antal sidor (inklusive denna sida)</i>	6
<i>Kursansvarig</i>	Kent Palmkvist, 1347, Kent.Palmkvist@liu.se
<i>Lärare som besöker skrivsalen</i> <i>Telefon under skrivtiden</i>	Kent Palmkvist 1347, 013-281347
<i>Besöker skrivsalen</i>	Ca kl 9 och kl 11 (+/- 30 minuter)
<i>Kursadministratör</i>	
<i>Tillåtna hjälpmedel</i>	Inga
<i>Betygsgränser</i>	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
<i>Visning</i>	Fredag 5 November 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).

Mikrokodsminnet 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, 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)	

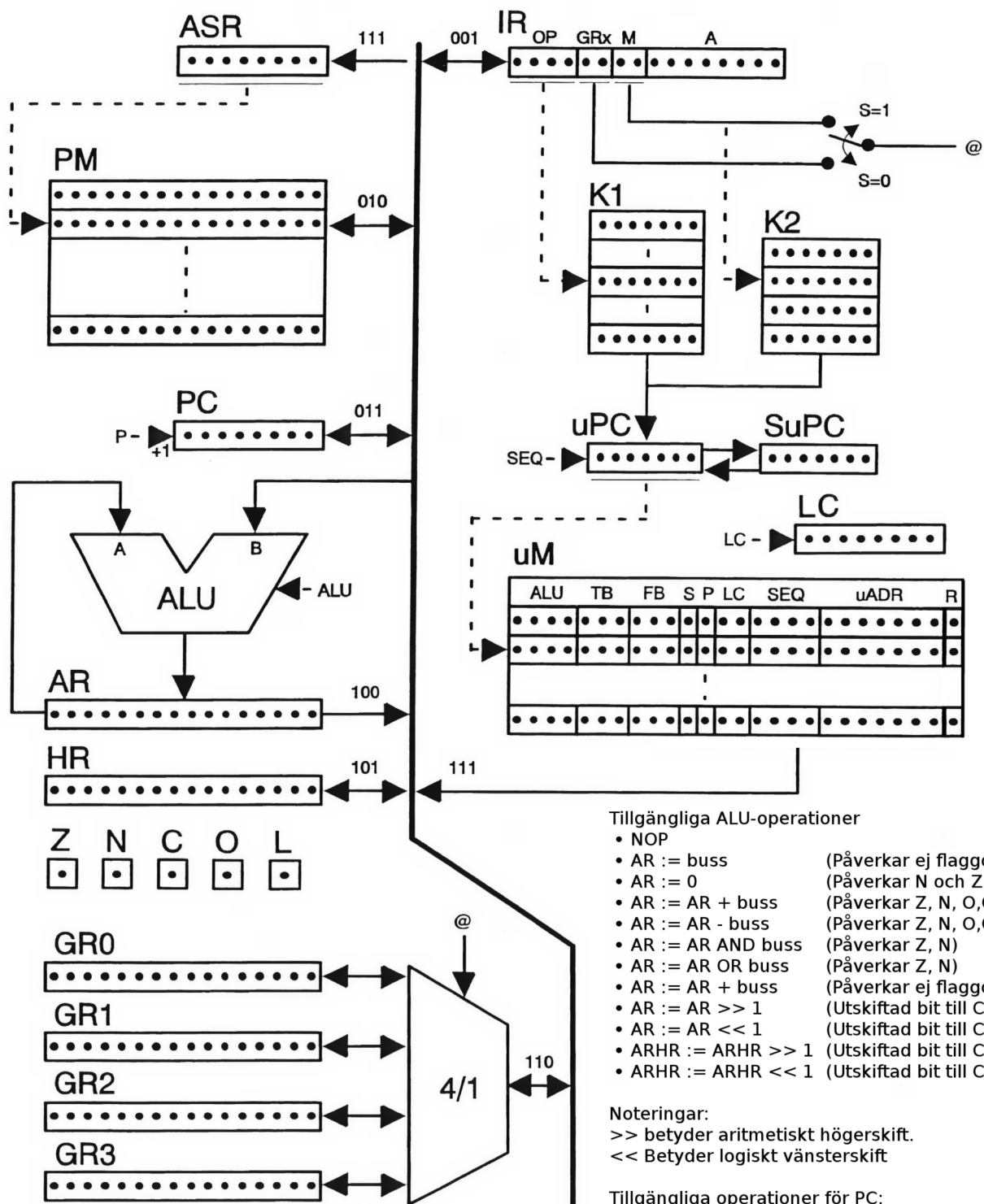
- (a) (8p) Skriv mikrokod för instruktionen MAXGR3 GR_x,M,A. Instruktionen ska jämföra GR_x med värdet i PM på den adress som definieras av adresseringsmode (alla adresseringsmoder ska stödjas), och det största av dessa två tal ska placeras i GR3. Indata antas vara 16-bitars tal i två-komplementsform.

Ange även adresserna för mikrokodsinstruktionerna som absoluta tal.

Exempel1: GR1 = 0x0123, PM(0x20)=0x4321. När instruktionen MAXGR3 GR1, 00,0x20 körs ska resultatet bli GR3 = 0x4321.

Exempel2: GR2 = 0x5678, PM(0x22)=0x0033, PM(0x33)=0x9809. När instruktionen MAXGR3 GR2, 10,0x22 körs ska resultatet bli GR3=0x5678.

- (b) (2p) Ange innehåll i K2 i binär form. Ange alla värden i binär form.



- 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) (2) Hur många jämförelser av tag-värde görs för varje läsning om en cache är en 4-vägs gruppassociativ cache med 1024 cachelines och 32 bytes cachelinelängd?
- (b) (2) Jämför egenskaper (pris, hastighet, komplexitet, volatilitet) mellan SRAM och DRAM.
- (c) (2) Skissa hur ett enkelt virtuellt minnessystem kan byggas upp om minnesarean är uppdelad i 16 lika stora minnesblock.
- (d) (2) Vilken typ av konflikt kan lösas genom data-forwarding?
- (e) (2) Vad är skillnaden mellan en processor av VLIW typ och en processor av SIMD typ?

Fråga 3: Assemblerprogrammering (10p)

I minnet ligger en tabell med 32-bitars tal lagrad, där bit 31-4 anger ett värde i tvåkomplementsform, och bit 3-0 (dvs minsta nibble) används som en typ av markering. Skriv en subrutin som hittar alla tal som har den markering som anges i register r2 och adderar dessa som 32-bitars värden (dvs tar bort markeringen och teckenförlänger värdet till 32 bitars längd). Tabellens start anges i register r0 och antal värden i tabellen anges i r1. Summan ska placeras i register r1 och därefter ska subrutinen Subrutin1 anropas. Subrutin1 definieras på annan plats och ska inte skrivas.

Exempel: r0 = 0x20001004, r1 = 0x00000006, r2 = 0x0000000a

Adress	Värde	Adress	Värde
0x20001000	0xabcdef01	0x20001010	0xabcd5542
0x20001004	0x0005abe1	0x20001014	0x0120006a
0x20001008	0xef01344a	0x20001018	0xe6432f01
0x2000100c	0x2345442a	0x2000101c	0x23642145

Subrutinen testar de 6 värdena på adresserna 0x20001004 – 0x20001018, identifierar att 2:a (0xef01344a), 3:e (0x2345442a) och 5:e (0x0120006a) värdet har rätt markering (högra nibble är a). För dessa värden med rätt markering tas markeringarna bort och varje värde teckenförlängs (0xfef01344, 0x02345442a, 0x00120006). Slutligen beräknas summan av dessa tre värden $0xfef01344 + 0x02345442 + 0x00120006 = 0x0136678c$. Subrutin1 anropas slutligen med r1=0x0136678c.

Fråga 4: Avbrott (8p)

En fläkt styrs av en avbrottsrutin. Skriv en avbrottsrutin som ska läsa ett 32-bitars värde som innehåller 8 sensorvärden från adress 0x40001000. Bit 3-0 är 1:a sensor, bit 7-4 är 2:a sensor etc. Avbrottsrutinen ska jämföra alla sensorvärden för att sedan skicka det största värdet (tolka de 4 bitarna från respektive sensor som ett positivt heltal) som ett 8-bitarsvärde på adressen 0x40001010. Bit 7-4 ska då vara noll. Om inga sensorvärden var större än 0xc ska avbrottet avslutas.

Om någon av sensorerna har ett värde större än 0x0c ska dessutom hela värdet som lästes från adressen 0x40001000 placeras i r5 och subrutinen Subrutin2 anropas och därefter ska värdet 0xff skickas ut på adress 0x40001020. Subrutin2 finns definierad på annan plats och ska inte skrivas.

Exempel: Avbrottsrutinen läser i en läsning det 32-bitars värdet 0x1E234568 från adress 0x40001000. Det största 4-bitarsvärdet är 0x0e och därför skriver avbrottsrutinen värdet 0x0e till adress 0x40001010. Eftersom värdet är större än 0x0c sätts register r5 till 0x1e234568 och subrutin2 anropas, och efter det skrivs värdet 0xff till adress 0x40001020.

Fråga 5: Aritmetik (6p)

- (a) (2p) Översätt decimaltalet -27 till ett 12-bitars tal i 2-komplementsform. Beskriv värdet som ett hexadecimalt tal.
- (b) (2p) Addera det 4-bitars 2-komplementstalet 1001_{2C} med 6-bitars positiva heltalet 10110_2 . Ställ upp den binära additionen och utför beräkningen. Svaret ska vara på formen av ett 8-bitars 2-komplements tal.
- (c) (2p) En 8-bitars processor utför additionen $0xAE+0x52$. Ange flaggornas värden (Z,C,N,O).

Fråga 6: Cache (6p)

En processor har en cache som har 64 byte långa cachelines. Adressbussen är 32 bitar lång. Cachen är en 4-vägs gruppassociativ cache. Cachens storlek är 32768 byte.

- (a) (2p) Hur många bitar används som index?
- (b) (2p) Hur mycket extra minne (räknat i antal bitar) finns i cache om man räknar bort kopiorna av data från primärminnet?
- (c) (2p) Antag cachen är tom. Hur många läsningar i sekvensen $0x12345678 + n*0x800$ ($n=0,1,2,3,\dots$) 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
```