Tentamen

Datorteknik Y, TSEA28

Datum	2025-01-10
Lokal	TER3, TERE
Tid	14-18
Utb. kod	TSEA28
Modul	DIT1
Kursnamn *Provnamn*	Datorteknik Y Digital tentamen
Institution	ISY
Antal frågor	6
Antal sidor (inklusive denna sida)	6
Kursansvarig	Kent Palmkvist, 1347, Kent.Palmkvist@liu.se
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Kent Palmkvist 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

Viktig information

- Hexadecimala värden anges som 0x1234
- 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

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) (8p) Skriv mikrokod för instruktionen BGT GRx,GRy,A (med M=GRy). Instruktionen ska jämföra värdet i GRx med värdet i GRy och göra ett relativt hopp ifall GRx > GRy, annars tas nästa instruktion. Addressen som i så fall skall hoppas till är PC+1+A (ifall GRy = 1 hoppas istället till PC+2+A). Värdena i GRx och GRy ska tolkas som 2-komplementsvärden. Instruktionen får påverka flaggorna.

Exempel: Antag GR1=0x0034, GR2=0xABCD. På adress 0x41 finns instruktionen BGT GR1, GR2, 0x12. Om denna körs resulterar detta i PC=0x54 eftersom 0x0034 > 0xABCD.

(b) (2p) Ange innehållet i minnet K2 i binär form. Antag att mikrokodsminnet har 128 rader. Ange både adress och data i K2 i binär form.

Fråga 2: Allmän teori (10p)

(a) (2p) Vad heter hårdvarustödet som möjliggör att två olika program få egna minnesceller trots att dom båda refererar till samma adresser i programmet. Förklara även kort hur dett är uppbyggt.

(b) (2p) Ange två skillnader mellan SRAM och DRAM?

(c) (2p) Vilken typ av konflikter löses med delayslots?

(d) (2p) Vad är en typisk egenskap för en processor av SIMD typ?

(e) (2p) Varför är cacheminnet viktigt i moderna datorer när det inte behövdes i datorer för 40 år sedan?

Fråga 3: Assemblerprogrammering (10p)

Skriv en subrutin som samlar ihop textdelar som lagrats på olika platser i minnet och placerar dom i en lång kontinuerlig sekvens i minnet, och sedan anropar en utskriftsrutin. Indata består av korta sekvenser av bytes i minnet, där första byten anger hur många byte av text som följer (positivt heltal, alltid större än 0), andra byten anger hur långt bort från starten av denna textdel som nästa textdel startar, och slutligen texten (1 tecken per byte). Avståndet ses som ett 2-komplementstal, och om avståndet är 0 betyder det att detta är sista textsdelen.

Efter att ha samlat ihop textdelarna till en kontinuerlig sekvens ska subrutinen Subrutin1 anropas, med r0= adress där den långa kontinuerliga sekvensen startar, och r1=antal tecken i den långa kontinuerliga sekvensen. Subrutin1 ska inte skrivas (antag den finns definierad på annan plats).

Vid anrop till subrutinen innehåller r0 adressen till start på 1:a korta textdelen, och r1 innehåller adressen där den långa kontinuerliga sekvensen ska starta. Bara minnesceller i stack och på platsen för den långa sekvensen får ändras av subrutinen.

Exempel: r0 = 0x20001002, r1 = 0x20001100, Minnesinnehåll:

Adress	Värden							
0x20001000	0x81	0x82	0x03	0x0E	0x48	0x65	0x6a	0x83
0x20001008	0x84	0x85	0x86	0x01	0x00	0x21	0x87	0x88
0x20001010	0x05	0xFB	0x20	0x48	0x6F	0x70	0x70	0x89

Första byte i första textdelen (0x03) anger att textdelen innehåller 3 tecken som ska kopieras, andra byte (0x0E) anger att nästa testdel startar på adress 0x20001002 + 0x0E = 0x20001010, och 3 sista byten i textdelen innehåller tecknen är 0x48, 0x65, 0x6a. Andra textdelen består av 5 tecken, och nästa textdel startar 5 adresser tidigare (dvs -5 = 0xFB) vilket ger startadress 0x20001010-5=0x2000100B och slutligen är de 5 tecknen 0x20, 0x48, 0x6F, 0x70, 0x70. Sista teckendelen startar på adress 0x2000100B och är 1 tecken långt, avståndsbyte = 0 säger att detta är sista textdelen, och tecknet som ska läggas till är 0x21. På adress 0x20001100 ska därför värdena 0x48, 0x65, 0x6a, 0x20, 0x48, 0x6F, 0x70, 0x70, 0x21 hamna. Subrutin 1 anropas sedan med r0=0x20001100 och r1=9.

Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som presenterar en summa av två värden. Vid varje avbrott ska ett 16-bitars ord först läsas från adress 0x40001010 (adress 0x40001012 får inte läsas). Om det inlästa värdet inte har både bit 13 = 0 och bit 15 = 1 så ska avbrottet avslutas. Om bit 13 = 0 och bit 15 = 1 i det inlästa värdet så ska två 5-bitars 2-komplementtal placerade i bit 9 – 5 respektive 4 – 0 konverteras till 32-bitars 2-komplementstal och summeras. Summan lagras som ett 32-bitars 2-komplementstal i r0. Därefter ska subrutinen Subrutin2 anropas och efter det ska summan beskriven som ett 16-bitars 2-komplementstal skrivas till adress 0x40001020 som en 16-bitar skrivning (adress 0x40001022 får ej skrivas till).

Subrutin2 förändrar värdet i register r4. Subrutin2 ska inte skrivas, utan antas vara definierad på annan plats.

Inga andra avbrott får starta medan denna avbrottsrutin kör.

Exempel: Avbrottsrutinen läser in värdet 0x9A27 från adress 0x40001010. Då bit 13=0 och bit 15=1 i detta värde summeras värdena från bit 5-0 = 00111_{2C} och bit 9-5 = 10001_{2C} och konverteras till 32-bitarstalen 0x00000007 och 0xFFFFFFF1 och summan 0xFFFFFFF8 placeras i r0. Därefter anropas Subrutin2 . Slutligen skrivs värdet 0xFFF8 i adress 0x40001020.

Fråga 5: Aritmetik (6p)

(a) (2p) Översätt det decimala värdet -9 till ett 6 bitars tvåkomplementstal.

(b) (2p) Antag summan 0x95 + 0x83 beräknas i en 8-bitars dator. Ange resulterande värden på flaggorna C, Z, N, V/O.

(c) (2p) Beräkna summan 10110111_{2C} + 1101_{2C} (båda är 2-komplementstal med 8 respektive 4 bitars ordlängd). Ange svaret som ett 10-bitars 2-komplementstal i binär form.

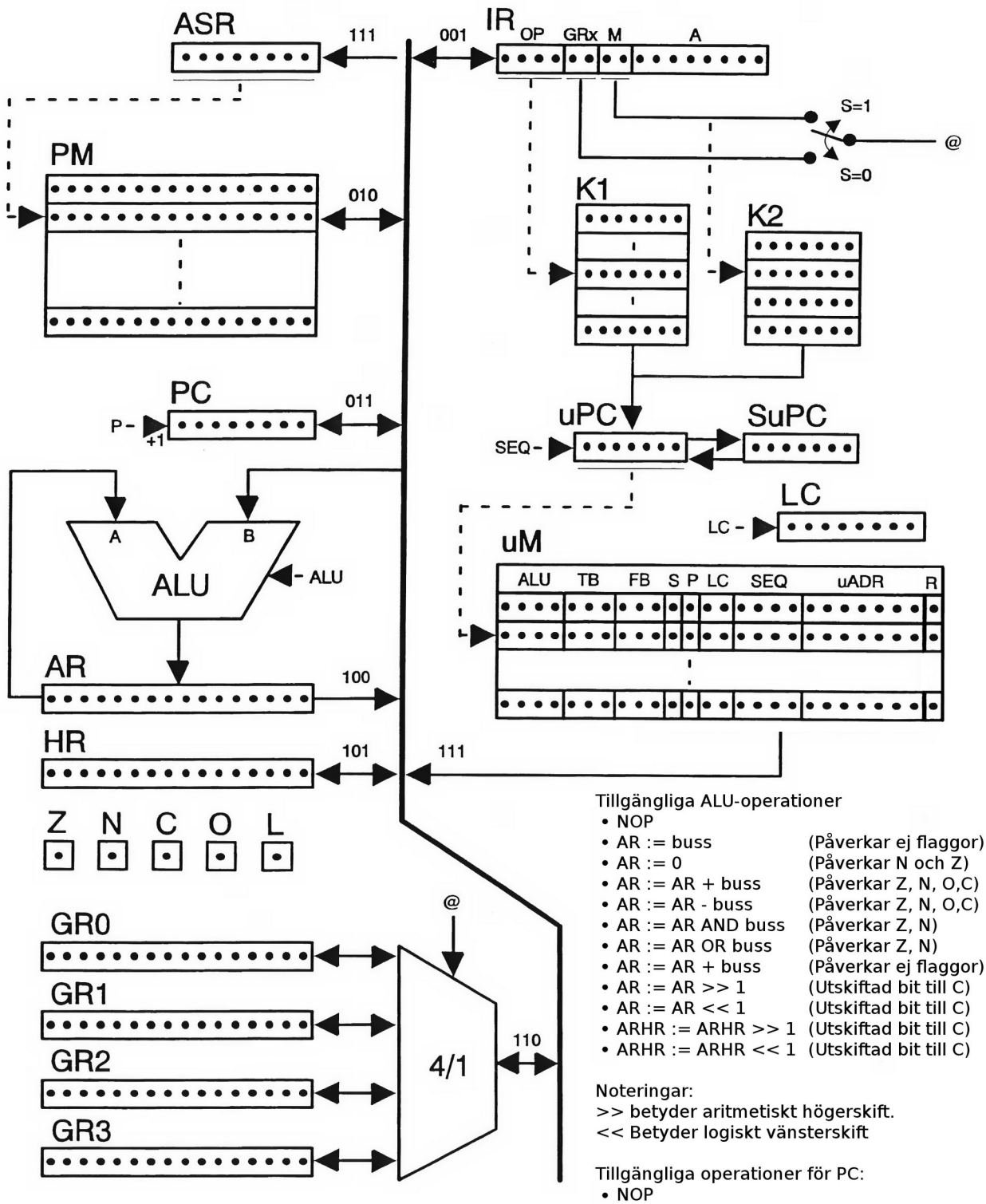
Fråga 6: Cache (6p)

En 2-vägs gruppassociativ cache är 32KB stor (32768 byte). Varje cacheline består av 8 byte. Adressbussen består av 32 bitar.

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

(b) (2p) Hur många bitar av adressen används som index?

(c) (2p) Antag cachen är tom. Ange för varje läsning om det är en cacheträff eller cachemiss om följande 8 adresser läses av processorn: 0xAB34567C, 0xAB345694, 0xAB34567E, 0xAB345684, 0x12345674, 0xAB34567A, 0x12345696, 0x1234568C. Antag varje läsning av processorn läser 1 byte.



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
BMI	Branch on minus	ROR	Rotate right
BNE	Branch on not equal	SBC,SBCS	Subtraction with C flag
BPL	Branch on plus	STR	Store register
BVC	Branch on overflow clear	STRB	Store register byte
BVS	Branch on overflow set	STRH	Store register halfword
BX	Branch and exchange	SUB,SUBS	Subtraction
CMP	Compare (Destination - Source)	TST	Test

```
; 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
```