Tentamen

Datorteknik Y, TSEA28

Datum	2022-08-16
Lokal	TER1
Tid	14-18
Kurskod	TSEA28
Provkod	TEN1
Kursnamn *Provnamn*	Datorteknik Y 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 *Telefon under skrivtiden*	Kent Palmkvist 1347, 013-281347
Besöker skrivsalen	Ca kl 15 och kl 17 (+/- 30 minuter)
Kursadministratör	
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Fredag 2/9 kl 13-14 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 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 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)	

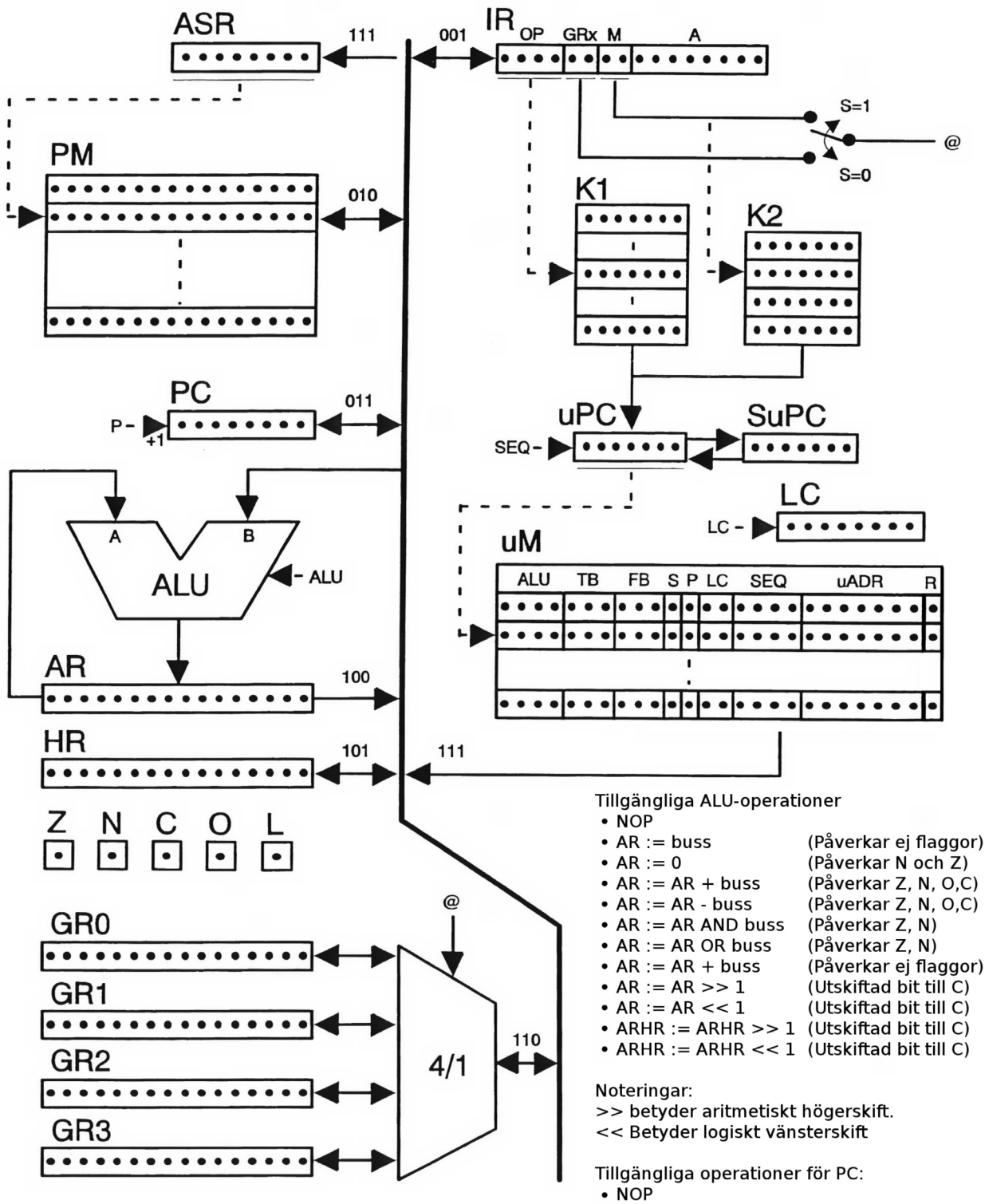
(a) (8p) Skriv mikrokod för instruktionen JSR M,A (subrutinanrop till programadress lagrad i minnesadress beskriven av M och A). GRx-bitarna i maskininstruktionen ska vara 00. Instruktionen ska spara återhoppsadressen till nästa instruktion på en stack i minnet. Stacken använder GR3 som pekar på nästa lediga minnesadress på stacken. Stacken växer mot lägre adresser.

Alla adresseringsmoder ska stödjas. Flaggor får påverkas. Ange adresserna för mikrokodsinstruktionerna som absoluta tal.

Exempel1: GR3 = 0x00A3, PM(0x20)=0x0042. När instruktionen JSR 00,0x20 på adress 0x12 körs ska resultatet bli PC=0x42, GR3=0x00A2, PM(0xA3)=0x0013.

Exempel2: GR3 = 0x00BE, PM(0x37)=0x0043. När instruktionen JSR 01,0x00 på adress 0x36 körs ska resultatet bli PC=0x43, GR3=0x00BD, PM(0xBE)=0x0038.

(b) (2p) Ange antal klockcykler (mikrokodssteg) det tar att utföra instruktionen JSR 10,0x22. Antag PM(0x22)=0x33, PM(0x33)=0x44, GR3=0xEA



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) Vad skiljer DRAM och SRAM från varandra?

(b) (2) Vad innebär egenskapen förhämtning (prefetch) hos en cache?

(c) (2) Vilken processor kan uföra flest instruktioner per sekund: En 4-vägs superskalär processor med 2 GHz klockfrekvens eller en 5-stegs RISC-processor med 3 GHz klockfrekvens? Motivera ditt svar.

(d) (2) Motivera varför det inte behövs två separata instruktioner för att kunna göra relativa hopp framåt respektive bakåt i ett program (t ex har lmia bara en BRA instruktion).

(e) (2) Vilket cacheminne innehåller flest minnesceller (dvs lagrar flest bitvärden): en 2 kilobyte direktadresserad cache eller en 2 kilobyte fullt associativ cache? Motivera ditt svar.

Fråga 3: Assemblerprogrammering (10p)

I minnet ligger en lista med 8-bitars positiva heltal lagrad. Skriv en subrutin som utför den inre loopen i bubbelsort. Subrutinen ska gå igenom listan en gång från första adress till näst sista och för varje adress jämföra värdet på adressen med värdet på nästa adress. Om värdet på nästa adress är mindre än värdet på nuvarande adress så ska värdena byta plats i minnet (dvs se till att det största av de två värdena är placerat i den högre adressen).

Listans start anges i register r0 och antal värden i listan anges i r1. När subrutinen är klar ska värdet i r1 ange om något par av värden har bytt plats. Register r1 ska sättas till värdet 1 om något värde bytt plats, annars ska värdet i r1 sättas till 0.

Exempel: r0 = 0x20001001, r1 = 0x00000005

Före subrutinanrop

Adress	Värde
0x20001000	0x36
0x20001001	0x17
0x20001002	0xb1
0x20001003	0x02
0x20001004	0x42
0x20001005	0xc2
0x20001006	0x22

Efter subrutinanrop

Adress	Värde
0x20001000	0x36
0x20001001	0x17
0x20001002	0x02
0x20001003	0x42
0x20001004	0xb1
0x20001005	0xc2
0x20001006	0x22

Subrutinen testar först värden på adress 0x20001001 och 0x20001002. 0x17 < 0xb1 så dom ligger rätt. Därefter jämförs värde på 0x20001002 och 0x20001003. Då 0xb1 > 0x02 så skrivs värdet 0x02 till adress 0x20001002 och 0xb1 till adress 0x20001003. Därefter jämförs värden på adresserna 0x20001003 och 0x20001004. 0xb1 > 0x42 så även här byter värdena plats. Slutligen jämförs adresserna 0x20001004 och 0x20001005 och där är 0xb1 < 0xc2 så inget byte görs. Innan subrutinen avslutas sätts r1 = 1 i detta exempel då minst 1 värde bytte plats.

Fråga 4: Avbrott (8p)

Ett par temperatursensorer läses av med ett timeravbrott (dvs i en avbrottsrutin). Skriv denna avbrottsrutin. Avbrottsrutinen ska läsa (med en läsning) ett 16-bitars värde som innehåller 2 sensorvärden från adress 0x40001010 (får t ex inte läsa från adress 0x40001012). Bitarna 11-6 är första sensorns värde (positivt heltal) och bitarna 5-0 är 2:a sensorns värde (positivt heltal).

Om skillnaden (absolutvärde) mellan dessa två sensorvärden är större än 5 ska subrutinen Subrutin2 anropas.

Innan avbrottet avslutas ska slutligen det största värdet av sensorerna skrivas som ett 8-bitars värde till adress 0x40001000 (får inte skriva till t ex 0x40001001).

Subrutin2 förstör register r4 och r7. Subrutin2 finns definierad på annan plats och ska inte skrivas.

Exempel: Avbrottsrutinen läser i en läsning det 16-bitars värdet 0x4568 från adress 0x40001010. Det första sensorvärdet är då 0x15 och den andra sensorns värde är 0x28. Då absolutvärdet hos skillnaden är 0x28-0x15 = 0x13 så är den större än 5. Därför anropas subrutinen Subrutin2. Slutligen skrivs värdet 0x0x28 till adress 0x40001000 innan avbrottet avslutas.

Fråga 5: Aritmetik (6p)

(a) (2p) Beräkna additionen av det binära 2-komplementsvärdet 11011_{2C} med det positiva heltalet 0x17. Beskriv svaret som ett 7-bitars 2-komplementstal i binär form.

(b) (2p) En 16-bitars processor utför additionen 0x7123+0x924C. Ange flaggornas värden (Z,C,N,O)

(c) (2p) Byt tecken på det positiva heltalet 0xce. Svaret ska anges som ett 12-bitars 2-komplementstal på binär form.

Fråga 6: Cache (6p)

En processor har en 2-vägs gruppassociativ cache som har 32 byte långa cachelines. Adressbussen är 24 bitar lång. Cachens storlek är 16384 byte.

(a) (2p) Bestäm hur många bitar som sparas i tag för varje cacheline.

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

(c) (2p) Antag cachen är tom. Hur många läsningar av adresserna enligt sekvensen 0x1234567 + n*0xc00 (n=0,1,2,...) kan göras innan cacheinnehåll kastas 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
```