Tentamen

Datorteknik Y, TSEA28

Datum	2020-10-21
Lokal	TER1, TER2, TERE
Tid	8-12
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 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	Onsdag 4 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).

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)	

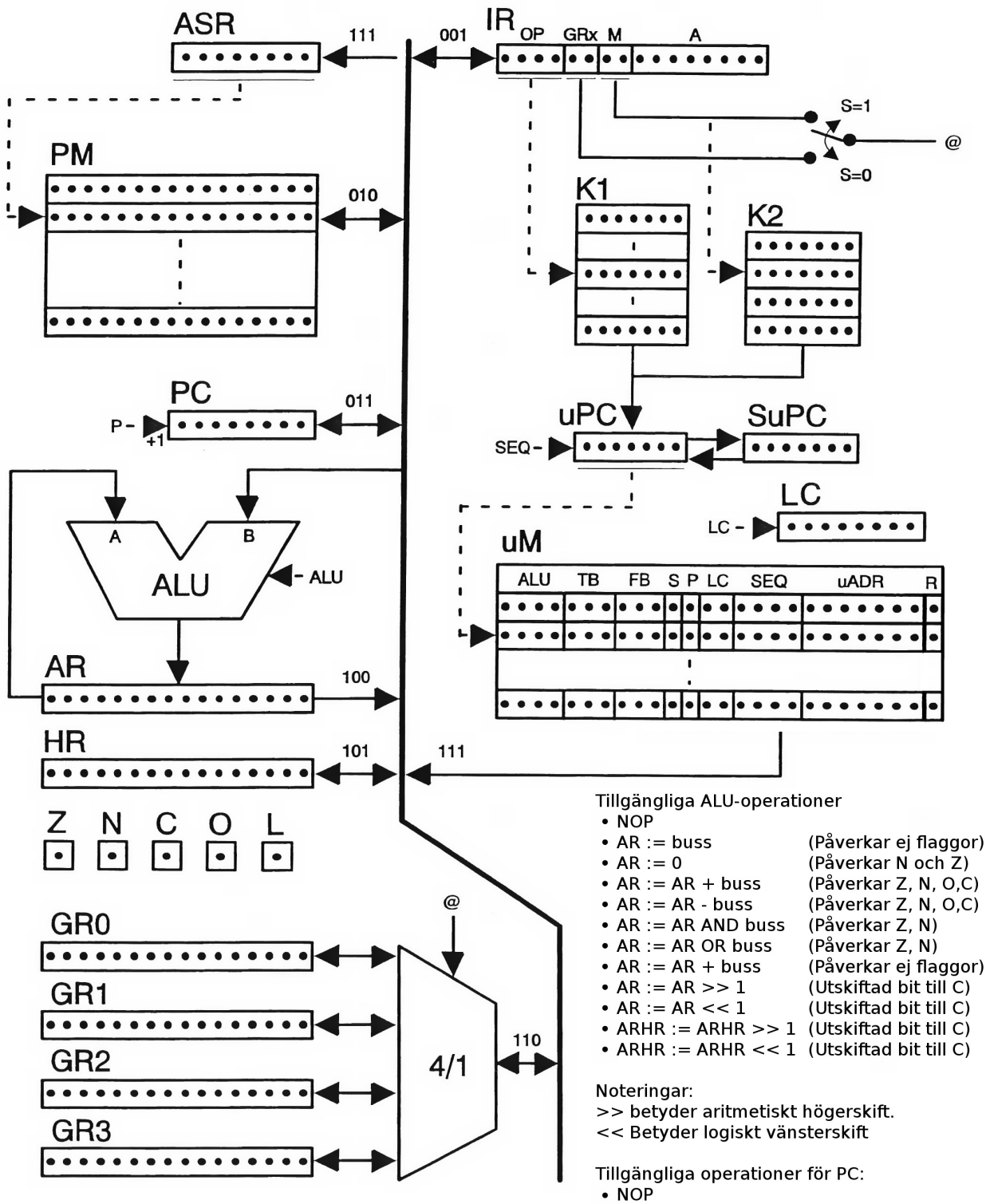
(a) (8p) Skriv mikrokod för instruktionen CMPBLT GRx,GRy,A. Instruktionen ska jämföra GRx med värdet i GRy (definierad av M-bitarna) och om GRx är strikt mindre än värdet GRy ska ett relativt hopp göras. Adressen hoppet ska ske till är instruktionens adress i minnet + A + 1. Indata i GRx och GRy antas vara i 2-komplementsform. Flaggorna C, N, Z och O får förstöras.

Ange även adresserna för mikrokodsinstruktionerna som absoluta tal.

Exempel1: GR1 = 0x0123, GR2 = 0x789A. Om instruktionen CMPBLT GR1, GR2,4 är placerad i minnesadress 6 blir resultatet ett hopp görs till adress 11 (0x0b).

Exempel2: GR2 = 0xfff1, GR0 = 0xfff9. Om instruktionen ~~CMPBLT GR2, GR0, 8~~ CMPBLT GR0,GR2,8 är placerad i minnesadress 4 blir resultatet att inget hopp görs.

(b) (2p) Ange innehåll i K2. Ange både adress och data 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) (2p) Kan en processor som inte är pipelinad få styrkonflikter? Motivera ditt svar.

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

(c) (2p) Varför behöver icke-volatilt minne finnas i en dator?

(d) (2p) Beskriv hur virtuellt minne kan hjälpa till att minska minnesbehov i en dator.

(e) (2p) Vilken typ av konflikt löser funktionaliteten register renaming? Inkludera ett exempel.

Fråga 3: Assemblerprogrammering (10p)

I minnet ligger en tabell med 16-bitars tal lagrad. Precis innan tabellens start ligger ett 16-bitars värde lagrat som anger hur många värden som används i tabellen. Skriv en rutin som lägger till ett nytt 16-bitars värde i tabellen på det index som anges, och räknar upp värdet som anger antal värden i tabellen. Detta betyder att alla tal från och med indexet måste flyttas nedåt ett steg i tabellen. Tabellens start anges i register r0, värdet som ska läggas till finns i register r1, och det index som ska användas finns i register r2. Index = 0 betyder att talet ska hamna längst upp i tabellen, index = 1 betyder att nya värdet ska läggas efter första värdet i tabellen etc.

Exempel: r0 = 0x20001004, r1 = 0x00001234, r2 = 0x00000003

Före subrutinanrop

Adress	Värde
0x20001000	0xabcd
0x20001002	0x0005
0x20001004	0xef01
0x20001006	0x2345
0x20001008	0x5678
0x2000100a	0x9abc
0x2000100c	0xa0b1
0x2000100e	0xc2d3
0x20001010	0xe4f5

Efter subrutinanrop

Adress	Värde
0x20001000	0xabcd
0x20001002	0x0006
0x20001004	0xef01
0x20001006	0x2345
0x20001008	0x5678
0x2000100a	0x1234
0x2000100c	0x9abc
0x2000100e	0xa0b1
0x20001010	0xe4f5

Subrutinen ökar värdet innan tabellen (adress 0x20001002) med ett och placerar det nya värdet (0x1234) som det fjärde värdet i tabellen (adress 0x2000100a) samt flyttar ned fjärde och femte värdet i den gamla tabellen till adress 0x2000100c respektive 0x2000100e. Inga andra värden i minnet ska ändras.

Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som ska läsa ett 32-bitars värde från en I/O-port på adressen 0x40001000. Denna läsning måste göras som en enkild instruktion pga I/O-portens konstruktion. Därefter ska en subrutin kallad Subrutin1 anropas med register r5 innehållande det inlästa värdet i omvänd nibbelordning. Dvs värdets nibblar (4-bitarsvärden) kommer i omvänd ordning (bit 3-0 placeras som bit 31-28, bit 7-4 som bit 27-24 etc., se exemplet nedan). Subrutinen kallad Subrutin1 kommer dessutom förstöra värdet i register r4 och r5. Subrutinen antas finnas definierad på annan plats och ska inte skrivas. Inga andra avbrott får startas medan avbrottsrutinen körs.

Exempel: Avbrottet läser i en läsning det 32-bitars värdet 0x12345678 från adress 0x40001000. Ett subrutinanrop görs sedan till Subrutin1 med register r5 = 0x87654321.

Fråga 5: Aritmetik (6p)

(a) (2p) Översätt decimaltalet -9 till ett 6-bitars binärt tal i 2-komplementsform.

(b) (2p) Antag en 8-bitars dator beräknar summan av decimaltalen 94 och 65. Ange summan i binär form och värdet på flaggorna Z, N, C och O.

(c) (2p) Vilken matematisk beräkning görs om ett 2-komplementstal skiftas 3 bitpositioner till vänster?

Fråga 6: Cache (6p)

En processor har en cache som har 32 byte långa cachelines. Adressbussen är 32 bitar lång. Cachen är en 8-vägs gruppassociativ cache. Index är 7 bitar.

(a) (2p) Hur många byte av data från primärminnet kan maximalt lagras i cacheminnet?

(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*0x100 (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
```