Tentamen

Datorteknik Y, TSEA28

Datum	2020-06-01
Lokal	Hemtenta
Tid	14-18
Kurskod	TSEA28
Provkod	TEN1
Kursnamn *Provnamn*	Datorteknik Y Skriftlig hemtentamen
Institution	ISY
Antal frågor	6
Antal sidor (inklusive denna sida)	6
Kursansvarig	Kent Palmkvist, Kent.Palmkvist@liu.se
Telefon/email under skrivtiden	Kent Palmkvist 013-281347 Kent.Palmkvist@liu.se
Tillåtna hjälpmedel	All kursmaterial. Ingen kontakt med annan person än examinator tillåten.
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	

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
- Svar inlämnas via lisams kursrum senast 18.15 Måndag 1 Juni 2020. Om problem uppstår med Lisam skickas motsvarande filer via email till Kent.Palmkvist@liu.se senast 18.20 Måndag 1 Juni 2020.

Lycka till!

Fråga 1: Mikroprogrammering (12p)

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 (128 olika adresser) 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)	

Instruktionerna STORE (opcode 6) börjar på mikrokodsadress 0x30, LOAD (opcode 2) på mikrokodsadress 0x35, BNE (opcode 9) på mikrokodsadress 0x38, ADD (opcode 8) på mikrokodsadress 0x42.

(8) Skriv mikrokod för instruktionen SWAPMAX GRx,M,A. Opcode = 5. Addresseringsmoder M = 00, 10 och 11 ska stödjas (mode 01 behöver inte hanteras alls). Instruktionen ska jämföra GRx med ett värde lagrat i minnet och spara det största i GRx och det mindre i minnet. Indata antas vara i 2-komplementsform. Flaggorna C, N, Z och O ska sättas i enlighet med resultatet av subtraktion GRx-minne.

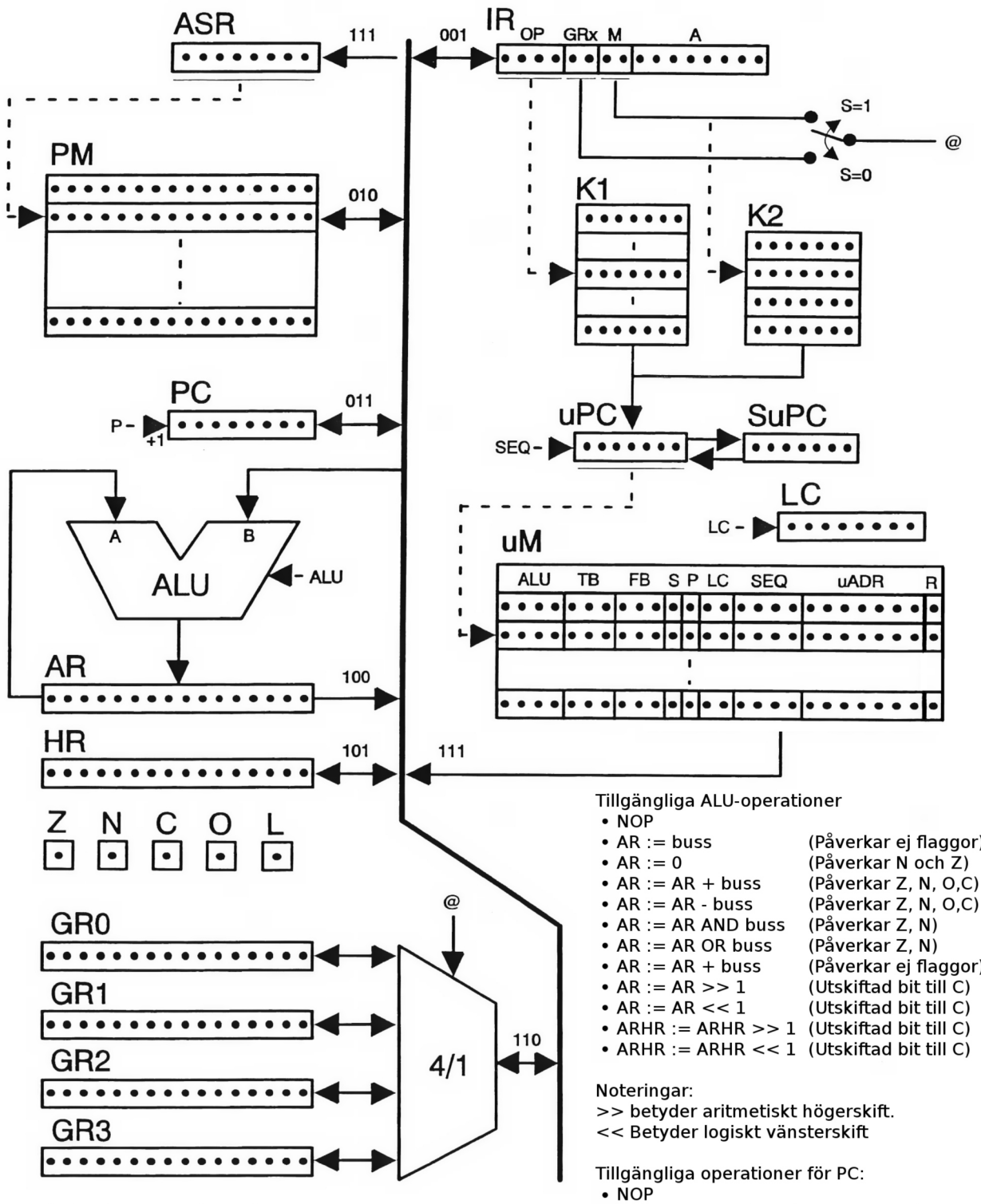
Ange även adresserna för mikrokodsinstruktionerna.

Exempel1: GR1 = 0x0123, PM(0x56) = 0x789A. Om ADDSAT GR1,0,0x56 utförs blir resultatet GR1 = 0x789A och PM(0x56) = 0x0123. Flaggorna Z=0, C=0, N=1, O=0.

Exempel2: GR0 = 0x9f01, PM(0x78) = 0x789A. Om ADDSAT GR0,0,0x78 utförs blir resultatet GR0 = 0x789A och PM(0x78) = 0x9f01. Flaggorna Z=0, C=1, N=0, O=1.

(2) Fyll i K1. Innehåll anges i binär form (både data och adress). Opcode är 4 bitar lång.

(2) Hur många klockcykler tar instruktionen SWAPMAX GR2, M=00, 0x40 att utföra om GR2=0xa123, PM(0x40)=0x6543?



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 (8p)

(a) (2p) Motivera varför DRAM är mer lämpat att användas i en dator med cache än i en dator utan cache?

(b) (2p) Lagras index-bitarna av en adress i minnesceller i en cache? Motivera ditt svar.

(c) (2p) Vad är för och nackdel med automatisk stackhantering vid subrutinanrop jämfört med användande av länkregister?

(d) (2p) Påverkar data-forwarding problemet med styrkonflikter? Motivera ditt svar.

Fråga 3: Assemblerprogrammering (8p)

I minnet finns en stor tabell lagrad. Varje rad i tabellen består av ett 16-bitars värde i 2-komplementsform samt ett 16-bitars radnummer till nästa rad som ska användas. Sista raden som ska användas har ett radnummer som har värdet 0.

Skriv en subrutin som adderar ihop de olika värdena i tabellen med start på första elementet i tabellen (rad nummer 0). Summan ska vara på 32-bitar format. Vid anropet till subrutinen innehåller register r0 adressen där tabellen startar och summan ska finnas i r1 när subrutinen returnerar. Tabellen ligger alltid på en adress som är jämnt delbar med 4.

Exempel: r0 = 0x20001004, minnesdump byte för byte

Adress	Värde	Adress	Värde
0x20001000	0x11 0x11 0x22 0x22	0x20001010	0x77 0xf7 0x02 0x00
0x20001004	0x48 0x65 0x03 0x00	0x20001014	0x55 0x55 0x66 0x66
0x20001008	0x68 0x00 0x70 0x70	0x20001018	0x76 0x84 0x00 0x00
0x2000100c	0x21 0x0a 0x05 0x00	0x2000101c	0x01 0x02 0x04 0x08

Subrutinen kommer läsa värdet för rad 0 vilket är 0x6548 (little endian), sedan läsa rad 3 eftersom radnumret är 0x0003, och då lägga ihop 0xfffff777 (2-komplement). Därefter läses rad 2 och slutligen rad 5. Summan som beräknas är 0x6548+0xfffff777+0x0a21+0xffff8476 = 0xffffeb56. Så när subrutinen returnerar ska r1 var 0xffffeb56.

Tabellen i exemplet som startar på adress 0x20001004 kan även ses som

Radnummer	Värde (16 bit)	Nästa Radnummer (16 bit)
0	0x6548	0x0003
1	0x0068	0x7070
2	0x0a21	0x0005
3	0xf777	0x0002
4	0x5555	0x6666
5	0x8476	0x0000
6	0x0201	0x0804

Fråga 4: Avbrott (8p)

Ett avbrott från en timer ska läsa två 8-bitars portar (GPIOADATA på adress 0x400043fc respektive GPIOBDATA på adress 0x400053fc). Om värdet från GPIOADATA har fler bitar som är 1 än antal bitar som är 1 i värdet från GPIOBDATA så ska subrutin Sub1 anropas och sedan ska värdet från GPIOBDATA skicka ut på GPIOADATA. Om inte Sub1 anropas ska subrutinen Sub2 anropas och sedan ska värdet från GPIOADATA skickas ut på GPIOBDATA istället.

Sub1 och Sub2 antas finnas definierat på annan plats i koden. Subrutinerna Sub1 och Sub2 förstör register r4, r5 och r6.

Exempel: Antag port GPIOADATA har värde 0x6E (dvs 5 bitar som är 1) och port GPIOBDATA har värde 0x84 (dvs 2 bitar som är 1). Då ska subrutin Sub1 anropas och därefter ska värdet 0x84 skickas ut på GPIOADATA innan avbrottet avslutas.

Fråga 5: Aritmetik (8p)

(a) (2p) Skriv det decimala talet -25 som ett 6-bitars binärt tal i 2-komplementsform.

(b) (2p) Utför operationen 0x34-0x67 (8-bitars dataordlängd) genom att beräkna differensen som addition (A-B = A+(-B)). Ställ upp additionen som binär addition och summera för hand.

(c) (2p) Bestäm värdet för flaggorna C, N, O, Z för subtraktionen 0x34-0x67 i uppgift b) ovan. Ange svaret i hexadecimal form.

(d) (2p) Bestäm svaret och flaggorna Z och N för operationen 0x34 AND 0x96 (bitvis logisk and). Antag det är en 8-bitars processor och att AND-instruktionen påverkar flaggorna.

Fråga 6: Cache (6p)

Ett program ska kopiera en del av en större bild från minnet. Varje pixel är 16 bitar stor. Startpunkt för en rad av 640 pixlar som ska kopieras är slumpmässig inom den stora bilden. Pixlarna på en rad ligger lagrade sekvensiellt i minnet, dvs nästa pixel ligger på adressen direkt efter föregående pixel.

Processorn som ska kopiera bilden har en tom cache när bildkopieringen börjar. Cachen är 128KByte stor (128*1024 byte) gruppassociativ cache. Cachelinelängd är 128 byte. Processorn har 32-bitars adress. Till index används 8 bitar.

(a) (2p) Hur många cachemissar pga läsning ger kopieringen av en rad om 640 pixlar?

(b) (2p) Hur många bitar av adressen används till tag?

(c) (2p) Hur många vägar finns i 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
```