Tentamen

Datorteknik Y, TSEA28

Datum	2022-10-19
Lokal	TER1
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 2/10 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 ABS GRx, M,A. Instruktionen beräknar absolutbeloppet av värdet i minnesadressen som definieras av M och A och placerar detta absolutbelopp i register GRx.

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

Exempel1: PM(0x20)=0xf542. När instruktionen ABS GR2,00,0x20 körs ska resultatet bli GR2=0x0abe.

Exempel2: GR3=0x37, PM(0x3c)=0x7542. När instruktionen ABS GR1,11,5 körs ska resultatet bli GR1 = 0x7542.

(b) (2p) Ange antal klockcykler (mikrokodssteg) det tar att utföra instruktionen ABS GR2, 10, 0x22. Antag PM(0x22)=0x33, PM(0x33)=0xf344.

Fråga 2: Allmän teori (10p)

(a) (2) Vad skiljer ett PROM och SRAM från varandra?

(b) (2) Varför är en buss som klarar burstöverföringar mer lämpligt att användas i en dator med cache jämfört med en dator utan cache?

(c) (2) Motivera varför en processor behöver kunna stänga av avbrott med hjälp av assemblerinstruktioner.

(d) (2) Kan en avbrottsrutin avgöra vilken instruktion som kördes när avbrottet skedde? Motivera ditt svar.

(e) (2) Varför sänks prestandan (antal utförda instruktioner per sekund) mer hos en pipelinad processor än hos en mikrokodad processor om många villkorliga hopp börjar utföras i ett assemblerprogram?

Fråga 3: Assemblerprogrammering (10p)

I minnet ligger en sekvens av värden lagrade i en tabell som 12-bitars heltal på 2-komplementsform plus ett 4-bitars värde som anger vilken rad som innehåller nästa värde i sekvensen. Första raden i tabellen är rad 0.

Skriv en subrutin som går igenom sekvensen av värden i tabellen och hittar det första lokala maximumet. Ett lokalt maximum har hittats om värdena före och efter det aktuella värdet båda är mindre än det aktuella värdet. Antag att minst ett lokalt maximum alltid existerar i sekvensen och att det aldrig är det första eller sista värdet i sekvensen.

Tabellens start anges i register r0. När subrutinen är klar ska värdet i r1 ange det lokala maximumets värde, och register r2 anger vilken rad i tabellen värdet låg på.

Exempel: r0 = 0x20001002, r1=0x00000006

Före subrutinanrop

Adress	Värde
0x20001000	0x0136
0x20001002	0xff17
0x20001004	0x2340
0x20001006	0x0234
0x20001008	0x104b
0x2000100a	0xc567
0x2000100c	0x0231

Efter subrutinanrop

Adress	Värde
0x2000100e	0x89ab
0x20001010	0xff93
0x20001012	0xcdef
0x20001014	0x0123
0x20001016	0x4567
0x20001018	0x1345
0x2000101a	0x89ab

Subrutinen kommer gå igenom tabellen med start på rad 0 på adressen 0x20001002, och sedan raderna 7, 3, 11, 5, 1 (adresserna 0x20001010, 0x20001008, 0x20001018, 0x2000100c, 0x20001004). De 12-bitar 2-komplementsvärdena på respektive rad i sekvensen är 0xff1, (dvs -15_{10}), 0xff9, 0x104, 0x134, 0x023, 0x234. Det första lokala maximumet är värdet 0x134 som finns på rad 11 i tabellen (värdena före och efter är 0x104 respektive 0x023 som båda är mindre än 0x134). Subrutinen returnerar med r1=0x00000134 och r2=0x0000000b.

Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som läser ett 32-bitars värde från adress 0x40001008 (får bara läsas en gång). Om bitarna 11-8 har fler ettor än bitarna 3-0 ska subrutinen Subrutin1 anropas. Slutligen ska bitarna 31-24 av det lästa 32-bitars värdet skrivas som ett 8-bitars värde till adress 0x40001010.

Subrutin1 förstör register r5 och r8. Subrutin1 finns definierad på annan plats och ska inte skrivas.

Exempel: Avbrottsrutinen läser i en läsning det 32-bitars värdet 0x7654abdc från adress 0x40001008. Bitarna 11-8 (0xb) innehåller 3 ettor, och bitarna 3-0 (0xc) innehåller 2 ettor. Eftersom bitarna 11-8 har fler ettor än bitarna 3-0 så anropas Subrutin1. Värdet 0x76 skrivs sedan som en 8-bitars skrivning till adress 0x40001010 och avbrottsrutinen avslutas sedan.

Fråga 5: Aritmetik (6p)

(a) (2p) Instruktionen subs r0,r1 görs (dvs r0-r1). Resulterande flaggvärden blir då N=0, C=1, Z=0, O=1. Ange vilket register som var störst om indata antas vara i 2-komplementsform. Ange även vilken flagga eller vilka flaggor som används för att bestämma detta.

(b) (2p) Ställ upp den binära additionen av de två 2-komplementstalen 11001_{2C} och 1010_{2C}. Utför sedan additionen och ange summan. Summan ska anges med minsta möjliga antal bitar som fortfarande kan beskriva summan korrekt som ett 2-komplementstal.

(c) (2p) Byt tecken på det positiva heltalet 23. Svaret ska anges som ett 7-bitars 2-komplementstal på binär form.

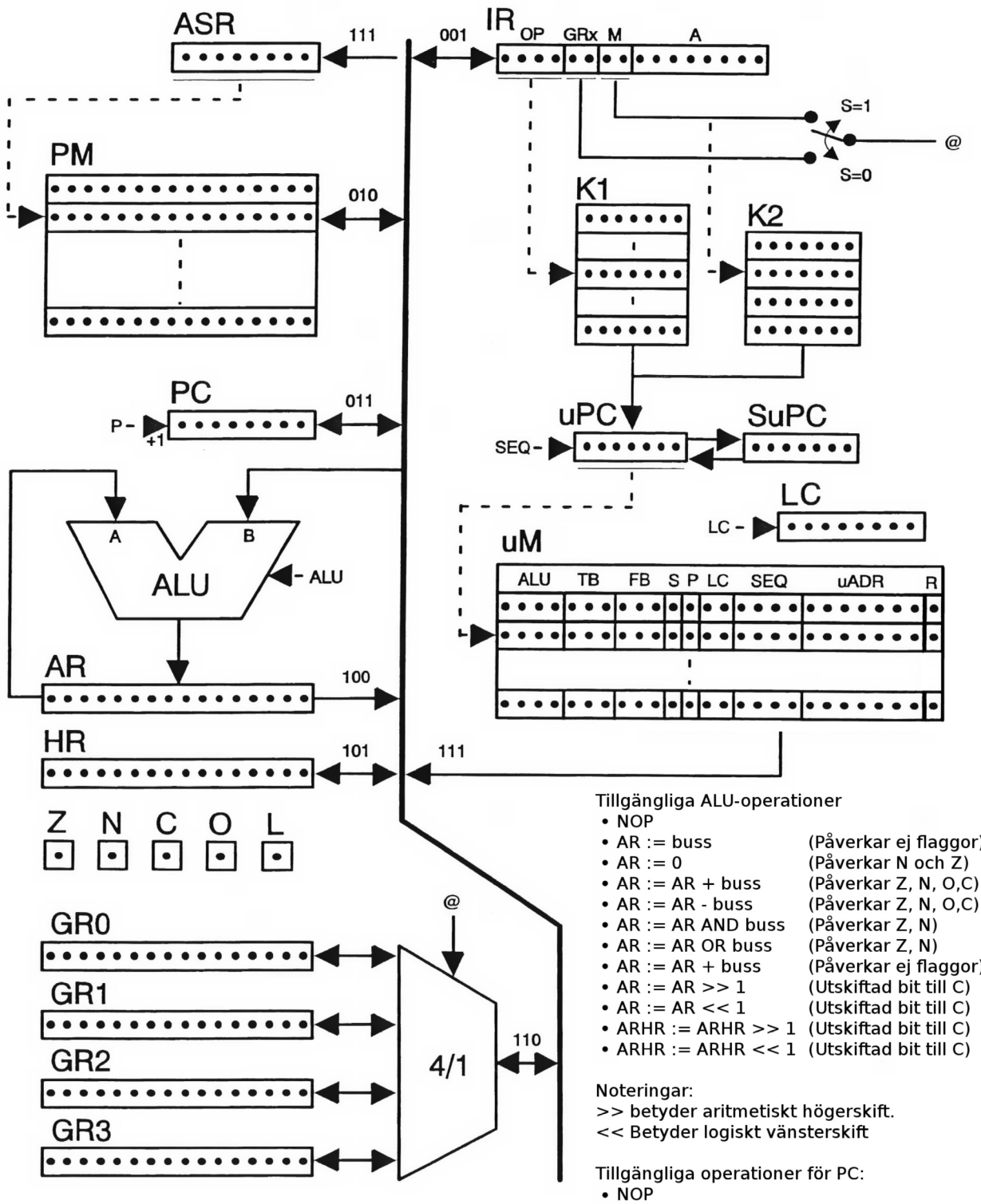
Fråga 6: Cache (6p)

En processor har en gruppassociativ cache som är 16384 byte stor. Adressbussen är 16 bitar lång. Varje cacheline är 8 byte lång. 9 bitar används som index av cachen.

(a) (2p) Bestäm hur många vägar cachen har (dess associativitet).

(b) (2p) Hur många adressbitar lagras i varje cacheline (räkna ej med data som lästs från minnet)?

(c) (2p) Antag cachen är tom. Hur cachemissar fås totalt om 32-bitars värden läses från minnet på adresserna 0x1234, 0x2230, 0x1230 samt 0x1238?



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 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
```