Tentamen (Exempel)

Datorteknik Y, TSEA28

Datum	2019-05-28
Lokal	G32, TER1, TER3, TER4, TERE, TERF
Tid	14-18
Kurskod	TSEA28
Provkod	TEN1
Kursnamn *Provnamn*	Datorteknik Y Skriftlig tentamen
Institution	ISY
Antal frågor	6
Antal sidor (inklusive denna sida)	7
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	Eva Zurawski
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Måndag 17 Juni 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)	

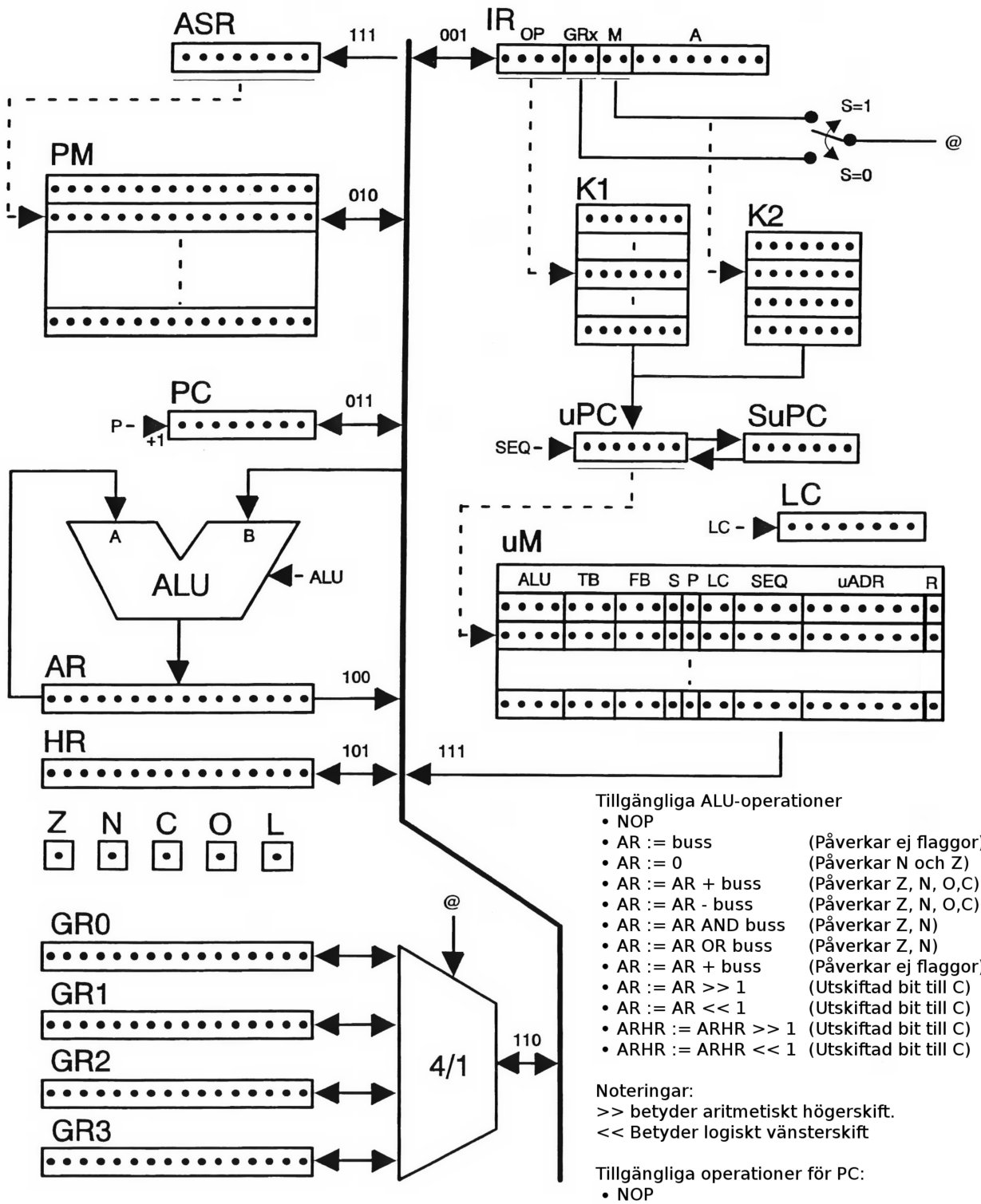
a) (8p) Skriv mikrokod för instruktionen CMPSKIPEQ GRx,M,A. Alla addresseringsmoder ska stödjas, och flaggorna kan ändras av instruktionen. Instruktionen ska jämföra GRx med ett värde lagrat i minnet. Om dessa värden är lika ska PC ökas med 1, dvs en instruktion ska hoppas över. Den överhoppade instruktionen får inte använda omedelbar adressering. Om värdena är lika utförs istället nästa instruktion direkt efter CMPSKIPEQ. OP-code för CMPSKIPEQ är 0xD. Mikrokodsminnet innehåller 128 rader-

Ange även adresserna för mikrokodsinstruktionerna.

Exempel: I PM(0x12) ligger instruktionen CMPSKIPEQ GR2,00,0xE1, och GR2=0x1234 och PM(0xE1)=0x1234. När instruktionen CMPSKIPEQ GR2,00,0xE1 körts ska nästa instruktion som utförs vara den som startar på adress 0x14. Detta då värdet i minnet var samma som i GR2. Instruktionen på adress 0x13 kommer därför inte utföras, utan ett hopp 1 steg framåt har gjorts.

b) (1p) Fyll i K2 i binär form. Ange både adress och data i binär form.

c) (1p) Ange det hexadecimala värdet i PM(0x12) för exemplet ovan, dvs maskininstruktionen i hexadecimal form för instruktionen CMPSKIPEQ GR2,00,0xE1.



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) Utför en pipelinad processor alltid 1 instruktion per klockcykel? Motivera ditt svar.

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

(c) (2p) Kan en additionsinstruktion resultera i flaggvärdena Z=1 och N=1 fås samtidigt? Motivera ditt svar.

(d) (2p) Vad innebär data forwarding hos en pipelinad processor?

(e) (2p) Ange två egenskaper för SRAM.

Fråga 3: Assemblerprogrammering (8p)

Skriv en subrutin som letar efter största värdet i en vektor bestående av 32 bitars ord. Det största värdet ska returneras i r0 och dess index (dvs relativa position i vektorn) ska returneras i r1. Startpositionen anges i register r0, och antal ord att undersöka anges i register r1. Värdet i ett i ord är beskrivet som ett 2-komplementstal. Första elementet i vektorn har index 0 och r1 har ett värde större än 0 vid start.

För 68000-kod innehåller A0 startpositionen, och D0 innehåller antal ord. Största värdet ska returneras i D0 och dess index i D1.

Exempel: r0 = 0x20001004, r1 = 0x00000007

Adress	Värde	Adress	Värde
0x20001000	0x11112222	0x20001018	0x77778888
0x20001004	0x48656a20	0x2000101c	0x55556666
0x20001008	0x686f7070	0x20001020	0x76541234
0x2000100c	0x210a0000	0x20001024	0x01020408
0x20001010	0xabcd0123	0x20001028	0xababcdcd
0x20001014	0x20001020	0x2000102c	0x23324444

När subrutinen är klar är register r0 = 0x77778888 eftersom det är störst av värdena 0x48656a20, 0x686f7070, 0x210a0000, 0xabcd0123, 0x20001020, 0x77778888, 0x55556666. Register r1 = 5 eftersom värdet är det 6:e värdet i vektorn.

Fråga 4: Avbrott (8p)

En avbrottrutin ska skrivas, som först anropar en subrutin kallad Subrutin1 som är definierad på annan plats i koden. Subrutin1 förstör registervärden i register r4 och r5. Subrutin1 returnerar ett värde i r0. De 4 minst signfikanta bitarna i det returnerade värdet i r0 (bit 3 – bit 0) ska sedan skrivas till minnesadress 0x40001238. De övriga bitarna på adress 0x40001238 (bit 7 – bit 4) får inte förändras av denna skrivning. När skrivningen till minnet är klar ska avbrottet avslutas.

För 68000 gäller att D0 används istället för r0, och att D1 och D2 påverkas av subrutinen Subrutin1.

Exempel: Värdet i minnesadress 0x40001238 är innan avbrottet 0x2a. Vid avbrottet anropas Subrutin1 som returnerar värdet 0x1234567 i r0. Värdet i minnesadress 0x40001238 ska efter avbrottet ha värdet 0x27.

Fråga 5: Aritmetik (8p)

(a) (2p) Skriv det decimala talet -5 som ett 6-bitars binärt tal i 2-komplementsform.

(b) (2p) Beräkna produkten av 0xC med 0xB. Indata är positiva heltal. Ange svaret i hexadecimal form.

(c) (2p) En 16-bitars dator beräknar summan 0x6223+0xbdac. Bestäm flaggornas värde (flaggor Z, C, N och O).

(d) (2p) Beräkna additionen av de två 2-komplementstalen ”10101” och ”0110101”. Ange svaret i decimal form.

Fråga 6: Cache (6p)

En av processorerna i Huaweis Kirin-chip har en instruktionscache som är 128 KB (131072 bytes) stor. Cachelinelängden är 64 byte. Denna cache är en 4-vägs gruppassociativ cache. Adressbussen är 64 bitar lång.

(a) (2p) Hur många rader finns det i en väg hos cachen?

(b) (2p) Hur många bitar består tag-fältet av?

(c) (2p) Antag cachen är tom. Hur många läsningar i sekvensen 0x40000000 + n*0x1000 kan göras innan inläst data börjar kastas ut ur cachen?

Kort repetition av M68000

Mnemonic	Kort förklaring	Mnemonic	Kort förklaring
ADD	Addition	EXT	Sign extend
ADDX	Add with X flag	EXTB	Sign extend a byte to 32 bit
AND	Logic and	JSR	Jump to subroutine
ASL	Arithmetic shift left	LEA	Load effective address
ASR	Arithmetic shift right	LSL	Logic shift left
BCC	Branch on carry clear	LSR	Logic shift right
BCS	Branch on carry set	MOVE	Move
BEQ	Branch on equal	MULS	Signed multiplication
BGT	Branch on greater than	MULU	Unsigned multiplication
BLT	Branch on less than	NEG	Negate
BNE	Branch on not equal	NOP	No operation
BRA	Branch always	NOT	Bitwise logic invert
BSR	Branch to subroutine	OR	Logic OR
CLR	Clear	ROL	Rotate left
CMP	Compare (Destination - Source)	ROR	Rotate right
DIVS	Signed division	RTE	Return from exception
DIVU	Unsigned division	RTS	Return from subroutine
EOR	Logic XOR	SUB	Subtract
EXG	Exchange	TST	Set integer condition codes

```
; Exempel på M68000 kod som kopierar 200 bytes från $2000 till $3000
    MOVE.L #$2000,A0
    MOVE.L #$3000,A1
    MOVE.B #50,D0
loop
    MOVE.L (A0)+,(A1)+
    ADD.B #-1,D0
    BNE loop
```

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
```