Tentamen (Exempel)

Datorteknik Y, TSEA28

Datum	2019-08-20
Lokal	TER1, TERE
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	Eva Zurawski, 6806, Eva.Zurawski@liu.se
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Onsdag 4 September 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)	

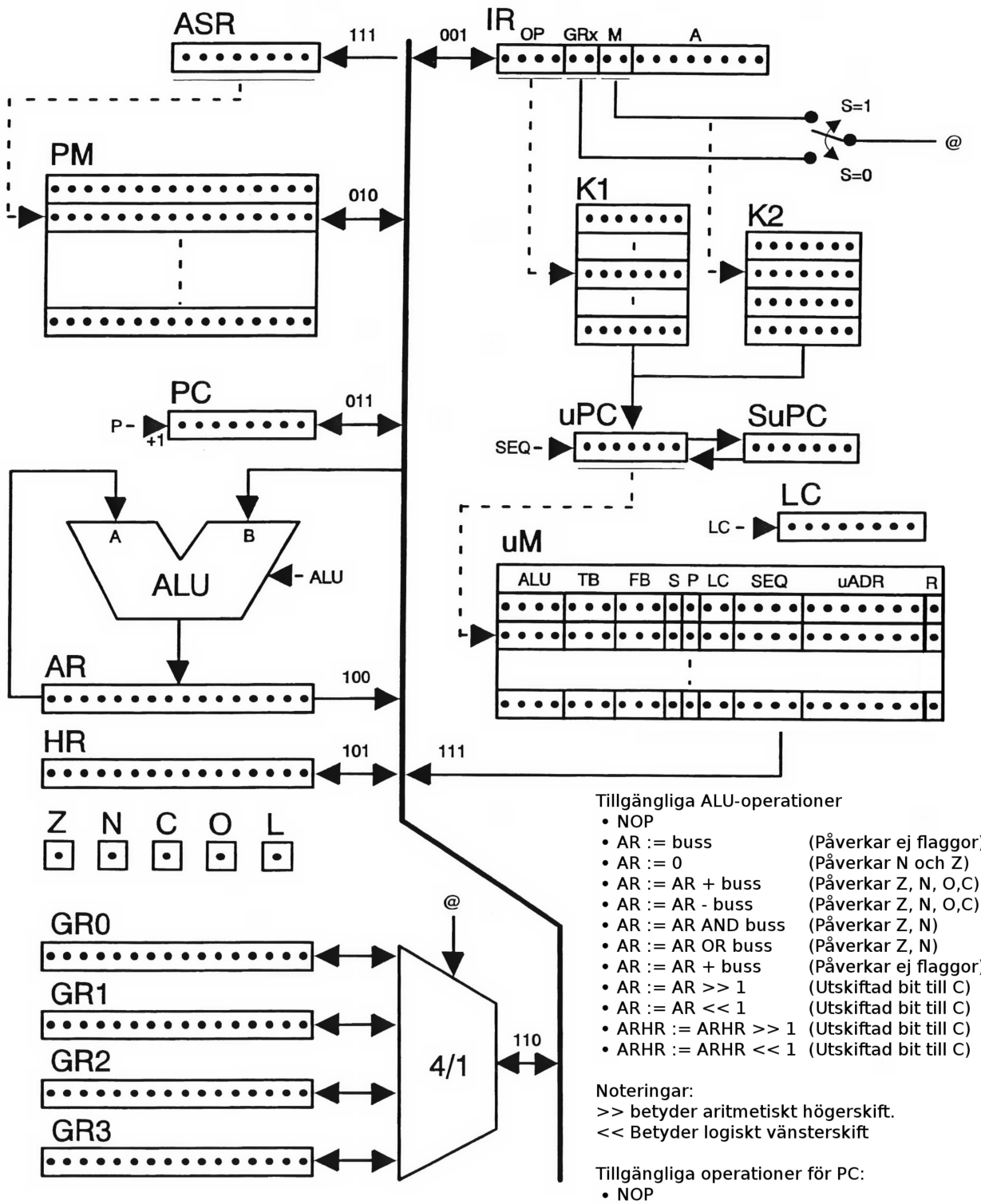
Skriv mikrokod för instruktionen ADDSAT GRx,M,A. Alla addresseringsmoder ska stödjas. Instruktionen ska addera GRx med ett värde lagrat i minnet och sparas i GRx. Indata antas vara i 2-komplementsform. Om additionen ger spill ska istället för summan ett av värdena 0x7fff och 0x8000 lagras i GRx, där 0x7fff väljs om summan är positiv, och 0x8000 väljs om summan är negativ. Flaggorna C, N, Z och O ska sättas i enlighet med resultatet av additionen innan hantering av spill gjorts.

Ange även adresserna för mikrokodsinstruktionerna.

Exempel1: GR1 = 0x0123, PM(0x56) = 0x789A. Om ADDSAT GR1,0,0x56 utförs blir resultatet GR1 = 0x79bd.

Exempel2: GR0 = 0x1234, PM(0x78) = 0x789A. Om ADDSAT GR0,0,0x78 utförs blir resultatet GR0 = 0x7fff.

Exempel3: GR2 = 0xfff1, PM(0x11) = 0xfff9. Om ADDSAT GR2,0,0x11 utförs blir resultatet GR2 = 0xffea.



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) Vad är skillnaden mellan datorsystem som är ”big endian” jämfört med ”little endian”?

(b) (2p) Vad skiljer en SIMD-processor från en VLIW-processor?

(c) (2p) Hur fungerar DMA?

(d) (2p) Motivera varför alla datorer har någon form av icke-volatilt minne.

(e) (2p) Är klockfrekvens ett bra mått på en dators prestanda? Motivera ditt svar.

Fråga 3: Assemblerprogrammering (8p)

I minnet ligger en text lagrad i ASCII-format. Skriv en subrutin som justerar tecknen så endast läsbara tecken skrivs ut. Denna justering består i att nollställa MSB (bit 7), och ersätta alla värden 0x00-0x1F samt 0x7F med värde 0x2E (ASCII för ’.’).

Subrutinen för utskrift heter printchar och är definierad på annan plats i koden. Subrutinen printchar förväntar sig ett tecken i r0, och förstör register r1-r3.

Vid anrop till din subrutin innehåller r0 adressen till 1:a tecknet och r1 innehåller antal tecken som ska skrivas ut.

Exempel: r0 = 0x20001004, r1 = 0x00000007

Adress	Värde	Adress	Värde
0x20001000	0x11 0x11 0x22 0x22	0x20001010	0x77 0x77 0x88 0x88
0x20001004	0x48 0x65 0x9a 0xc1	0x20001014	0x55 0x55 0x66 0x66
0x20001008	0x68 0x00 0x70 0x70	0x20001018	0x76 0x54 0x12 0x34
0x2000100c	0x21 0x0a 0x00 0x00	0x2000101c	0x01 0x02 0x04 0x08

Subrutinen kommer anropa printchar 7 gånger, och r0 har följande värden: 0x48, 0x65, 0x2e, 0x41, 0x68, 0x2e, 0x70.

Fråga 4: Avbrott (8p)

En reklamskylt styrs av en ARM-processor där portarna GPIOADATA och GPIOBDATA (8 bitar var) styr alla lampor som sitter runt skylten. För att få ljuset att rotera skapas ett avbrott var 200:e ms mha en timer. Rotationen ska skifta alla bitar ett steg åt vänster, och den mest signifikanta biten hos den ena porten skickas alltid till den minst signifikanta biten hos den andra porten.

Det finns en annan avbrottsrutin som ibland ändrar värdet på portarna GPIOADATA och GPIOBDATA, så din avbrottsrutin måste hindra den andra avbrottsrutinen från att påverka portarna medan din avbrottsrutin uppdaterar värden i GPIOADATA och GPIOBDATA. Adressen för GPIOADATA = 0x400043fc och GPIOBDATA = 0x400053fc.

Exempel: Antag port GPIOADATA har värde 0x6E och port GPIOBDATA har värde 0x84. Efter ett avbrott ska då port GPIOADATA ha värdet 0xDD, och port GPIOBDATA ha värdet 0x08.

Fråga 5: Aritmetik (6p)

(a) (2p) Skriv det decimala talet -11 som ett 7-bitars binärt tal i 2-komplementsform.

(b) (2p) Beräkna produkten av 0xC med 0xB. Indata är 4-bitars 2-komplementstal. Ange svaret i binär form.

(c) (2p) En 16-bitars dator beräknar summan 0x6227+0xbd9c. Vad blir resultatet? Ange svaret i hexadecimal form.

Fråga 6: Cache (8p)

Intels senaste processor Sunny Cove har en L2 datacache som är 512 Kbyte (524288 byte) stor. Den är 8-vägs gruppassociativ. Cachelinelängden är 64 byte. Adressbussen är 64 bitar lång.

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

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

(c) (2p) Hur mycket extra minne finns i cachen för att lagra tag?

(d) (2p) Antag cachen är tom. Hur många läsningar i sekvensen 0x20000000 + n*0x100000 kan göras innan 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
```