Tentamen (Exempel)

Datorteknik Y, TSEA28

Datum	2018-xx-xx
Lokal	TER1, TER3
Tid	8-12
Kurskod	TSEA28
Provkod	TEN1
Kursnamn *Provnamn*	Datorteknik Y Skriftlig tentamen
Institution	ISY
Antal frågor	7
Antal sidor (inklusive denna sida)	7
Kursansvarig	Kent Palmkvist, 1347, kentp@isy.liu.se
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Kent Palmkvist 1347, 013-281347
Besöker skrivsalen	Ca kl 9 och kl 11
Kursadministratör	Gunnel Hässler
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Måndag xx Xxxxxx kl 12.30 – 14.00 i kursansvarigs kontor

Viktig information

- Hexadecimala värden kan anges 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

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	PM := GRx, S := 0, R := 0, uPC := 0	; STORE
8	uPC := 0 om Z = 1, annars uPC++	; JNE addr
9	PC := IR, uPC := 0	
10	GRx := PM, S := 0, R := 0, uPC := 0	; LOAD

(a) (7p) Skriv mikrokod för instruktionen ADDLONG GRx,GRy,A (add long value GRxGRy + (A)(A+1)). Addera 32-bitarstalet som finns i GRx kombinerat med GRy (GRx är mest signifikant del, GRy anges i M-fältet) med 32-bitars talet placerat på adress A samt A+1 (på adress A finns mest signifikant del av värdet). Svaret ska placeras i GRx och GRy med mest signfikant del i GRx. Flaggorna N, C och O ska vara korrekta för additionen, medan Z får förstöras.

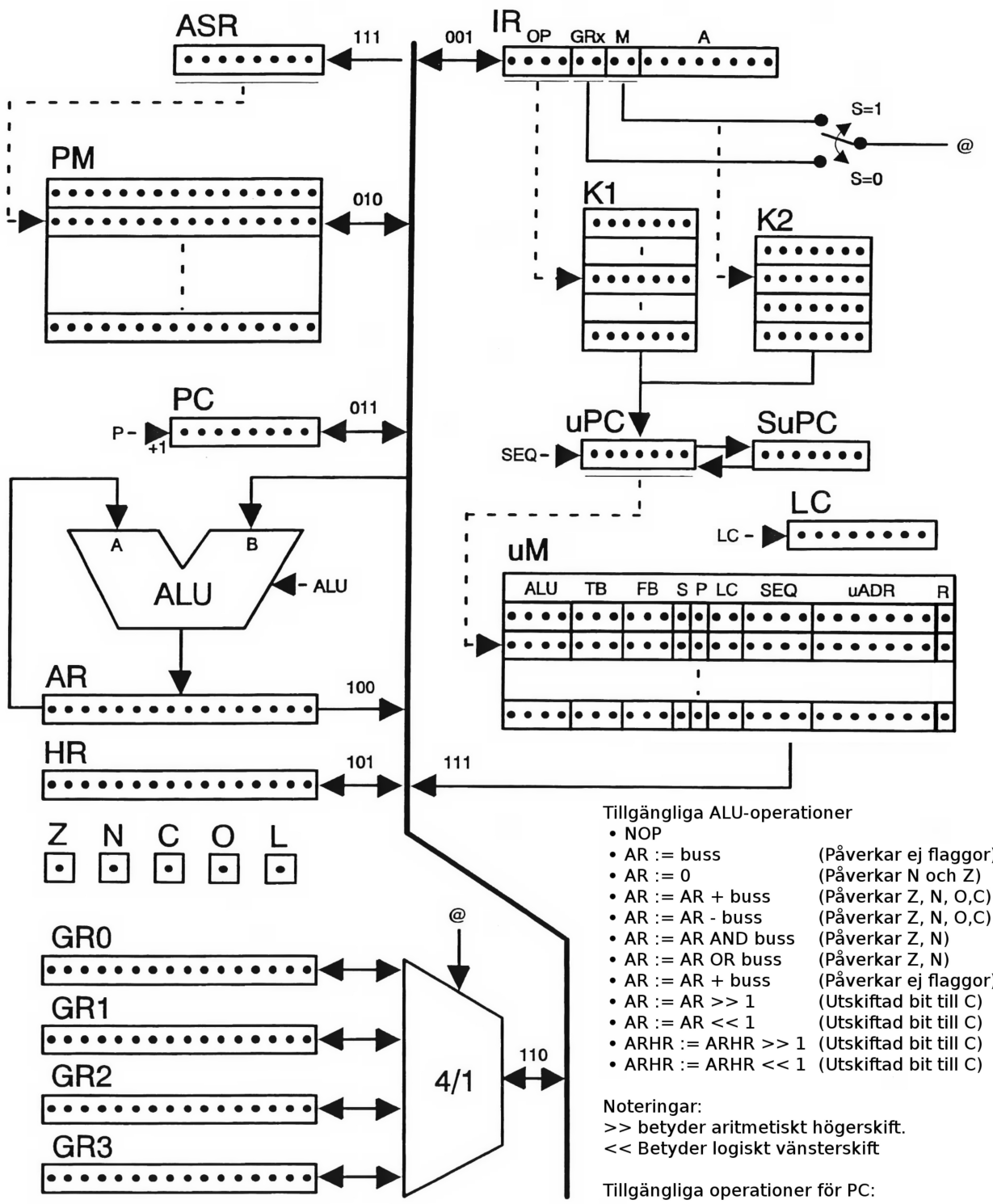
Ange även adresserna för mikrokodsinstruktionerna.

Opcode	Instruktion	Betydelse	Adresseringsmoder	Påverkar flaggor
0001	ADDLONG	GRxGRy = GRxGry+ (A)(A+1)	Anger GRy	C,Z,N,O

Exempel: GR1=0x12, GR2=0x34, PM(0x56)=78,PM(0x57)=E0. Efter ADDLONG GR1,GR2,0x56 ska 0x1234+0x78E0 (=0x8B14) beräknas vilket resulterar i GR1=0x8B, GR2=0x14. Flaggorna ska efter utförd instruktion vara C=0, O=1 och N=1.

(b) (2p) Hur många klockcykler (dvs steg) tar det att utföra instruktionen inklusive hämtning av instruktionen etc. Ange alla olika möjligheter.

(c) (1p) Ange bitmönstret i hexadecimal form för maskininstruktionen ADDLONG GR2,GR1,0x51.



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 skiljer en processor av VLIW-typ från en superskalär processor?

(b) (2p) Vad är en styrkonflikt i en datorarkitektur?

(c) (2p) Kan två olika datorarkitekturer (med olika antal interna bussar, register etc.) implementera samma instruktionsuppsättning? Motivera ditt svar.

(d) (2p) Vad är skillnaden mellan RAM och ROM?

(e) (2p) Vad är skillnaden mellan instruktionerna aritmetisk skift höger (ASR) och logisk skift höger (LSR)?

Fråga 3: Assemblerprogrammering (8p)

Skriv en subrutin som skickar ut värden (8 bitar stora) från minnet till en I/O-enhet. I/O-enheten indikerar om den kan ta emot ett nytt tecken genom att bit 1 i värdet på adress 0x10040 (8 bitar stort) är 0 om det går att skicka ett tecken. Tecken skickas genom skriva dess värde på adress 0x10042. Adressen till 1:a tecknet i minnet ska finnas i r0 (32-bitars värde) och antal tecken anges i r1 (16-bitars värde) när subrutinen anropas. Notera att bit 0 samt bit 2-7 på adress 0x10040 har andra funktioner.

För 68000 använd A0 istället för r0, D0 istället för r1

Exempel: r0 = 0x4000, r1 = 0x0003, 0x4000 = 0x12345678 => Skicka ut 0x12 när bit 1 på adress 0x10040 är 0, skicka ut 0x34 när bit 1 på adress 0x10040 är 0, och sedan 0x56 och 0x78 på samma sätt.

Fråga 4: Avbrott (10p)

(a) (8p) En avbrottsrutin ska skrivas som styr en 7-segments display ansluten på adress 0x10080. En tabell bestående av 4 byte finns lagrad på adress 0x5000-0x5003. Vilken byte som ska styra värdet att skicka ut på displayen bestäms av bit 1 och 0 på adress 0x10082. Notera att övriga bitar på adress 0x10082 kan ha okända värden. För att ge rätt utseende på displayen ska den byte som valts från adress 0x5000-0x5003 användas som index i tabellen av byte som startar på adress 0x5010. Värdena i tabell med start på adress 0x5000 är alltid mellan 0x00 och 0x0F. Tabellerna på adress 0x5000-0x5003 samt 0x5010-0x501F är redan definierade.

Exempel: Antag tabell på adress 0x5000 har värdena 0x02, 0x04, 0x06 och 0x08. Antag 0x10082 har värdet 0x1A, och tabell på adress 0x5010 har värden 0x20 till och med 0x2F: Avbrottsrutinen ska då läsa värdet på adress 0x5002 (eftersom bit 1 och 0 = ”10”), vilket betyder att 6:e elementet i tabellen på 0x5010 (0x5016=0x26) ska skickas till port 0x10080.

(b) (2p) Hur mycket utrymme måste finnas tillgängligt på stacken innan avbrottet sker för att avbrottsrutinen ska fungera (dvs hur många byte använder avbrottet)? Ange om du använder ARM Cortex-M eller 68000.

Fråga 5: Aritmetik (6p)

(a) (2p) Beräkna subtraktionen x-y om x är ett 3-bitars tvåkomplementtal ”110” och y är ett 5-bitars tvåkomplementtal ”01101”. Ange svaret i binär form som ett 5-bitars 2-komplementtal.

(b) (2p) Beskriv det decimala värdet 42 i hexadecimal form.

(c) (2p) Vad blir värdena på flaggorna C, O, Z, N efter en 8-bitars addition av de två hexadecimala talen 0xB3 och 0xAD?

Fråga 6: Cache (6p)

En processor med 32-bitars adress har en cache på 16 Kbyte (16384 bytes, dvs 2^14 bytes). Denna cache är en 2-vägs gruppassociativ cache med cachelinelängd av 16 byte. Ersättningsalgoritmen behåller det värde som lästs senast. Ett beräkningsprogram som körs i processorn läser minnet på adresserna enligt sekvensen: 0x3000+0x100*n, dvs 0x3000, 0x3100, 0x3200 etc. Antag att cache från början är tom.

(a) (2p) Hur många cachelines finns i cacheminnet totalt?

(b) (2p) Hur många bitar av adressen används som index i cacheminnet?

(c) (2p) Hur många läsningar hinner beräkningsprogrammet göra innan gamla värden börjar tas bort 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

addr1: .field 0x2000,32
addr2: .field 0x3000,32

main:  ldr  r0,addr1
       ldr  r1,addr2
       mov  r3,#50
loop:
       ldr  r4,[r0],#4
       str  r4,[r1],#4
       subs r3,r3,#1
       bne  loop
```