Tentamen (Exempel)

Datorteknik Y, TSEA28

Datum	2018-08-21
Lokal	TER4
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, kentp@isy.liu.se
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Kent Palmkvist 1347, 013-281347
Besöker skrivsalen	Ca kl 15 och kl 17
Kursadministratör	
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Onsdag 5 September kl 12.30 – 13.30 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 (9p)

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).

a) (8p) Skriv mikrokod för instruktionen BLKCPY GRx,GRy,A. GRy-registret definieras av M-bitarna i maskininstruktionen. Denna instruktion ska kopiera A stycken värden från PM, med start på adressen som GRx pekar på t o m adressen GRx+A-1, och placera dessa värden i PM på adresserna GRy t o m GRy+A-1. Endast de 8 lägsta bitarna i GRx respektive GRy används som adresser. Efter instruktionen ska värdet i GRx och GRy ökat med A. Flaggornas värden får förstöras.

Exempel: GR1 = 0x0034. GR2 = 0x0023, PM(0x34) = 0x8222. PM(0x35) = 1234, PM(0x36) = 0x5678. Efter BLKCPY GR1,GR2,3 är PM(0x23) = 0x8222, PM(0x24) = 0x1234, PM(0x25) = 0x5678 samt GR1 = 0x0037 och GR2 = 0x0026.

b) (1p) Hur många olika maskininstruktioner (OP-koder) kan finnas om OP-kodsfältet utökas till 10 bitar?

Fråga 2: Allmän teori (10p)

(a) (2p) Vad skiljer en fullt associativ cache från en grupp-associativ cache?

(b) (2p) Vad innebär det att man hanterar styrkonflikter med ”delayed branch”?

(c) (2p) Hur skiljer sig en superskalär processor från en VLIW-processor?

(d) (2p) Har alla processorer samma antal generella register? Motivera ditt svar.

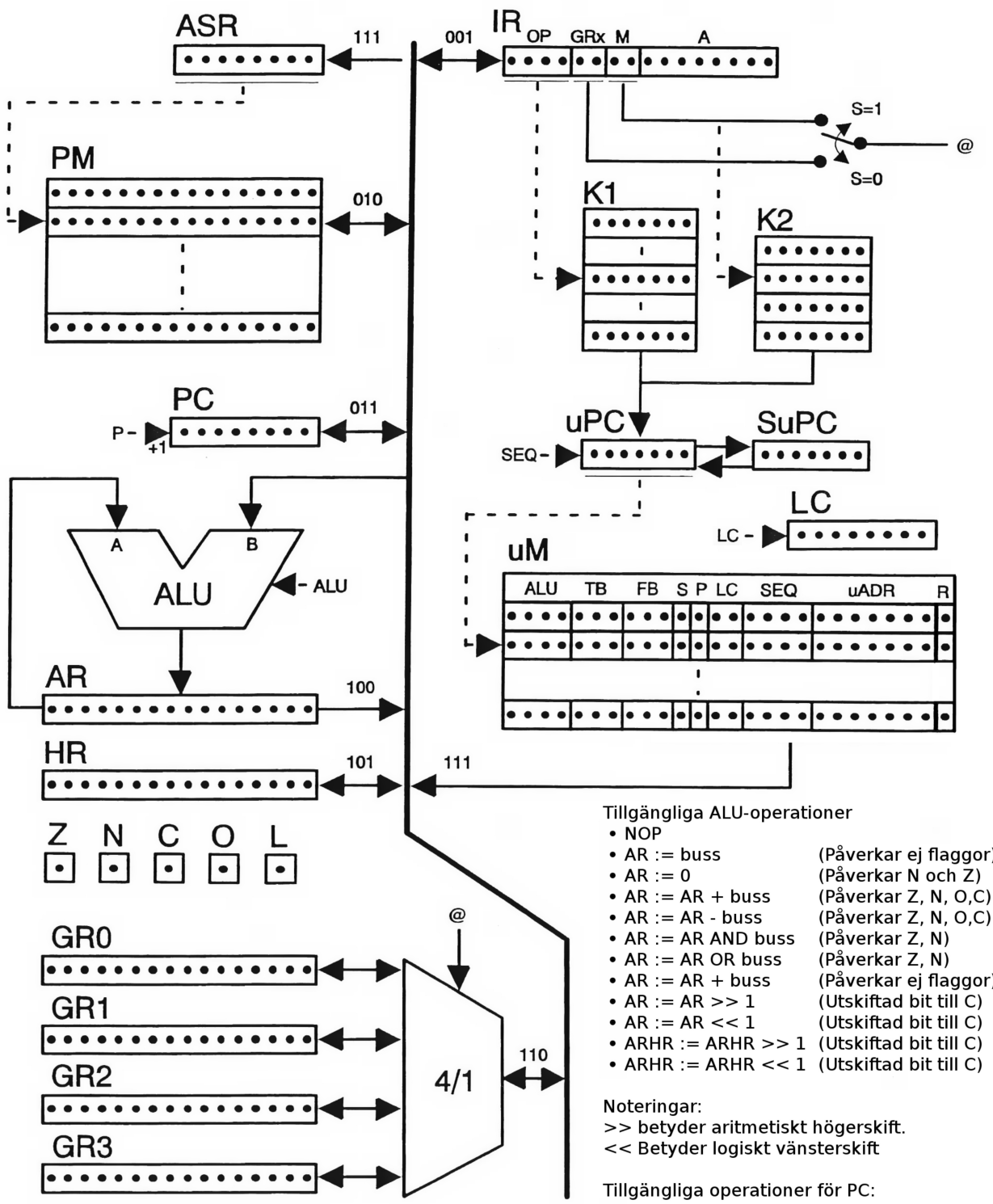
(e) (2p) Måste en processor innehålla mikrokod? Motivera ditt svar.

Fråga 3: Assemblerprogrammering (10p)

Skriv en subrutin som skriver ut det 32-bitars värdet i registret r0 i hexadecimal form (som ASCII-tecken). För att skriva ut tecken finns en subrutin på adress 0x01234568 som skickar ut ASCII-tecknet som finns i register r1 (men som inte ändrar några register). ASCII-tecknet för siffran ’0’ är värdet 0x30, siffran ’1’ har värdet 0x31, ... , siffran ’9’ har värdet 0x39, bokstaven ’A’ har värdet 0x41, ..., och bokstaven ’F’ har värdet 0x46.

För 68000 använd D0 istället för r0 och D1 istället för r1.

Exempel: r0 = 0x12AB349F. Subrutinen ska då anropa subrutinen på adress 0x01234568 8 gånger med r1 = 0x31, 0x32, 0x41, 0x42, 0x33, 0x34, 0x39 och slutligen 0x46.



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 4: Avbrott (8p)

En avbrottrutin ska skrivas, som läser en sektor från en hårddisk genom att göra 512 läsningar av 8-bitars värden från adress 0x401000C0. Dessa 8-bitars värden ska placeras efter varandra i minnet med start på den 32-bitars adress som finns angiven i minnet på adress 0x20001010. När avbrottet är klart ska 32-bitars värdet i minnet på adress 0x20001010 ha ökats med 512.

Inga andra avbrott får startas medan denna avbrottsrutin körs.

Exempel: Om värdet på adress 0x20001010 = 0x23456780 och första läsning av 0x401000C0 är 0x12, andra läsning av 0x401000C0 är 0x34, etc. så ska avbrottsrutinen skriva i minnet på adress 0x23456780 = 0x12, 0x23456781 = 0x34 etc. När avbrottet är klart ska 512 stycken 8-bitars värden ha sparats på adresserna 0x23456780 till och med 0x2345697F och värdet på adress 0x20001010 ska vara 0x23456980.

Fråga 5: Aritmetik (8p)

(a) (2p) Addera 7-bitars binära 2-komplementstalet ”0100110” med 6-bitars binära 2-komplementstalet ”110111”. Resultatet ska vara ett 8-bitars binärt tal på 2-komplements form.

(b) (2p) En 8-bitar processor beräknar summan 0xE3 + 0x86. Vilka värden får flaggorna Z, N, C, O.

(c) (2p) Översätt värdet -5 till ett 6-bitars binärt tal på 2-komplementsform.

(d) (2p) Vilka värden på flaggorna indikerar att resultatet av en subtraktion av två 2-komplementstal är negativt?

Fråga 6: Cache (5p)

En 32-bitars processor har en cache där varje cacheline är 32 byte lång. Denna cache är 2-vägs gruppassociativ. För index används 9 bitar.

En matris är lagrad i minnet och adresserna läses i sekvensen 0x4000, 0x4100, 0x4200, etc. Antag cacheminnet från början är tomt.

(a) (2p) Hur stor är cachen (i bytes)?

(b) (2p) Hur många läsningar av matrisen kan läsas innan data från första läsningen på adress 0x4000 tas bort ur cachen?

(c) (1p) Vad skiljer en write-through cache från en write-back cache?

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
```