Tentamen

Datorteknik Y, TSEA28

Datum	2016-08-16
Lokal	TER2, 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	Cirka kl 15 och 17
Kursadministratör	Gunnel Hässler
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Fredag 2 September kl 12.30 – 14.00 i kursansvarigs kontor

Viktig information

- 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 (11p)

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: Mikroprogrammerings-ordet 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) (7p) Implementera operationen ADD GRa,GRb. Operationen GRa := GRa + GRb ska beräknas där GRa och GRb är valfria register GR0-GR3. Opcode ska vara 1101, och GRa anges i GRx-fältet och GRb anges i M-fältet. A-fältet ska vara 0. Flaggorna Z,N,C, och O ska påverkas av instruktionen.

Exempel: Register GR0 innehåller värdet $7128, och register GR1 innehåller värdet $24A9. Efter instruktionen ADD GR0,GR1 ska GR1 fortfarande innehålla $24A9 och värdet i GR0 ska vara $7128+$24A9=$95D1. Flaggorna ska då få värdet C=0, N=1, O=1, Z=0.

(b) (1p) Ange bitmönstret för maskininstruktionen ADD GR2,GR1.

(c) (3p) Utöka instruktionen till att även addera värdet i A-fältet i instruktionen. Dvs instruktionen blir ADD GRa, GRb,A där A är en positiv konstant på 8 bitar. Operationen som beräknas blir GRa := GRa + GRb + A.

Exempel: ADD GR0,GR1,$93 med GR0 = $7128, GR1 = $24A9. Resultatet blir GR0 =$7128+$24A9+$93 = $9664 med C=0, N=1, O=1 och Z=0.

Fråga 2: Allmän teori (10p)

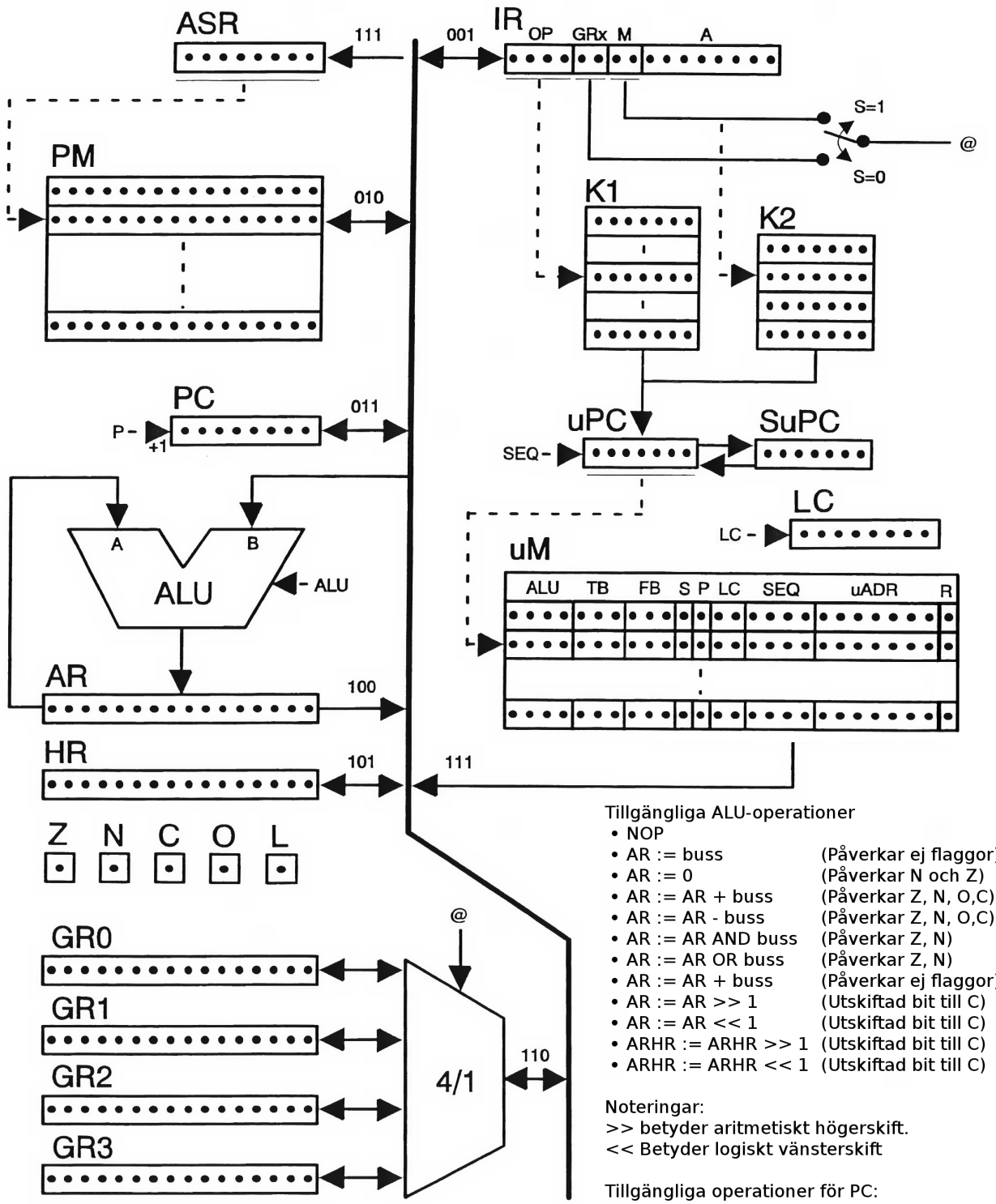
(a) (2p) Vad är skillnaden mellan en dator av typ ”big endian” jämfört med en dator av typ ”little endian”?

(b) (2p) Ange en fördel och en nackdel med att använda en PCI-buss istället för en enklare datorbuss (t ex den som finns i tutor-systemet).

(c) (2p) Beskriv hur en dator med virtuellt minne översätter virtuella adresser till fysiska adresser.

(d) (2p) Ange två typer av halvledarminnen som behåller informationen även vid spänningsbortfall.

(e) (2p) Vad skiljer aritmetisk skift åt höger (ASR) mot operationen logiskt skift åt höger (LSR)?



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 3: Aritmetik (5p)

(a) (2p) Talet -3 ska representeras med 8 bitar i 2-komplementsform. Ange bitmönstret.

(b) (3p) 2-komplementstalen $17 (5 bitar långt) och $C2 (8 bitar långt) ska adderas. Beräkna summans bitmönster (8-bitar långt 2-komplementstal).

Fråga 4: Assemblerprogrammering (8p)

I minnet ligger en text lagrad i ASCII-format. Skriv en subrutin som kopierar och justerar denna text så att den går att skriva ut utan att ge konstiga tecken på skärmen. MSB (bit 7) i varje tecken ska därför nollställas, och alla tecken (efter nollställning av MSB) med värdet $00-$1F och $7F ska ersättas med värdet $2E (ASCII för '.').

Vid anropet till subrutinen innehåller A0 adressen till 1:a tecknet i strängen, A1 innehåller adressen där justerade tecken ska sparas, och D0 innehåller antal tecken som ska kopieras och justeras.

Exempel: Subrutinen anropas med A0 = $00004022, A1=$00004034, D0=$00000009

Minnesadress	Värde innan	Minnesadress	Värde efter
$00004020	$123BC865	$00004020	$123BC865
$00004024	$6AA120BA	$00004024	$6AA120BA
$00004028	$9D290A12	$00004028	$9D290A12
:		:	
$00004034	$3456789A	$00004034	$48656A21
$00004038	$BCDEF012	$00004038	$203A2129
$0000403C	$3456789A	$0000403C	$2156789A

Fråga 5: Cache (8p)

En 128 kilobyte (2^17=131072 byte) stor cache ska anslutas till en processor med 32-bitars adressbuss. Denna cache ska vara 8-vägs associativ och varje cacheline består av 8 byte.

(a) (2p) Hur många cachelines finns per väg i cachen?

(b) (2p) Hur många bitar lagras i cacheminnet, inklusive tag?

(c) (3p) Hur lång sekvens kan läsas i minnet från en slumpmässig adress n (dvs läs adress n, n+1, n+2 etc.) innan ett tidigare värde ur sekvensen försvinner ur cachen. Ange både minsta och största antal möjliga läsningar innan ett tidigare värde kastas ut ur cachen.

(d) (1p) Hur många komparatorer (jämförare) finns det i cacheminnet, dvs hur många jämförelser görs vid varje minnesaccess?

Fråga 6: Avbrott (8p)

Skriv en avbrottsrutin som ska hantera mottagande av tecken från en I/O-enhet (t ex ett tangentbord) till en buffert i minnet. Denna avbrottsrutin startas både av en timer som ger avbrott varje millisekund samt när ett tecken tagits emot av I/O-enheten.

Avbrottet behöver inte nollställas (inga speciella adresser behöver läsas/skrivas). I/O-enheten har ett statusregister (8 bitar) på adress $10040 där bit 3 = 1 om ett tecken finns att ta emot. Om bit 3 = 0 ska avbrottsrutinen avslutas. Det finns även fler bitar till andra funktioner i statusregistret som inte ska användas i denna rutin. Tecken från I/O-enheten kan läsas på adress $10042 (8 bitar). Detta tecken ska lagras i en buffert.

I minnet på adress $5010 (8 bitar) finns ett värde som anger hur många lediga platser det finns i bufferten. Varje gång I/O-enheten har ett tecken att ta emot ska en koll göras om det finns plats i bufferten innan tecknet läses från I/O-enheten. Om det inte finns plats (värde = 0) ska rutinen avslutas. Om plats finns ska värdet på adress $5010 räknas ned med 1 varje gång ett tecken sparats i bufferten.

I minnet på adress $5000 (32 bitar) finns en adress lagrad som pekar på var i minnet tecknet från I/O-enheten ska lagras. Adressen lagrad i minnet på adress $5000 ska räknas upp varje gång ett tecken sparats i bufferten.

Exempel:

Om $5000=$00004020, $5010=5, och $10040 = $13: Läsning av statusregiter visar att bit 3 = 0, och avbrottsrutinen avslutar direkt. Värden i minnet på adress $5000 och $5010 påverkas inte.

Om $5000=$00004123, $5010=3, och $10040 = $24, $10042 = $41: avbrottsrutinen sätter minnesaddress $4123=$41, sätter $5000=$00004124, $5010=2.

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
```