Tentamen

Datorteknik Y, TSEA28

Datum	2015-06-01
Lokal	G32, G34, G35, G36, G37, TER3
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

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 (8p)

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) (1p) Förklara vad K1 respektive K2 används till?

(b) (6p) Skriv instruktionen STRLEN GRx. Den ska ge som svar i GRx det antal tecken som ingår i strängen som GRx pekar på och som termineras (avslutas) med värdet 0. Flaggorna får förändras fritt av instruktionen. De övriga GR-registren och PC samt innehållet i PM får inte påverkas.

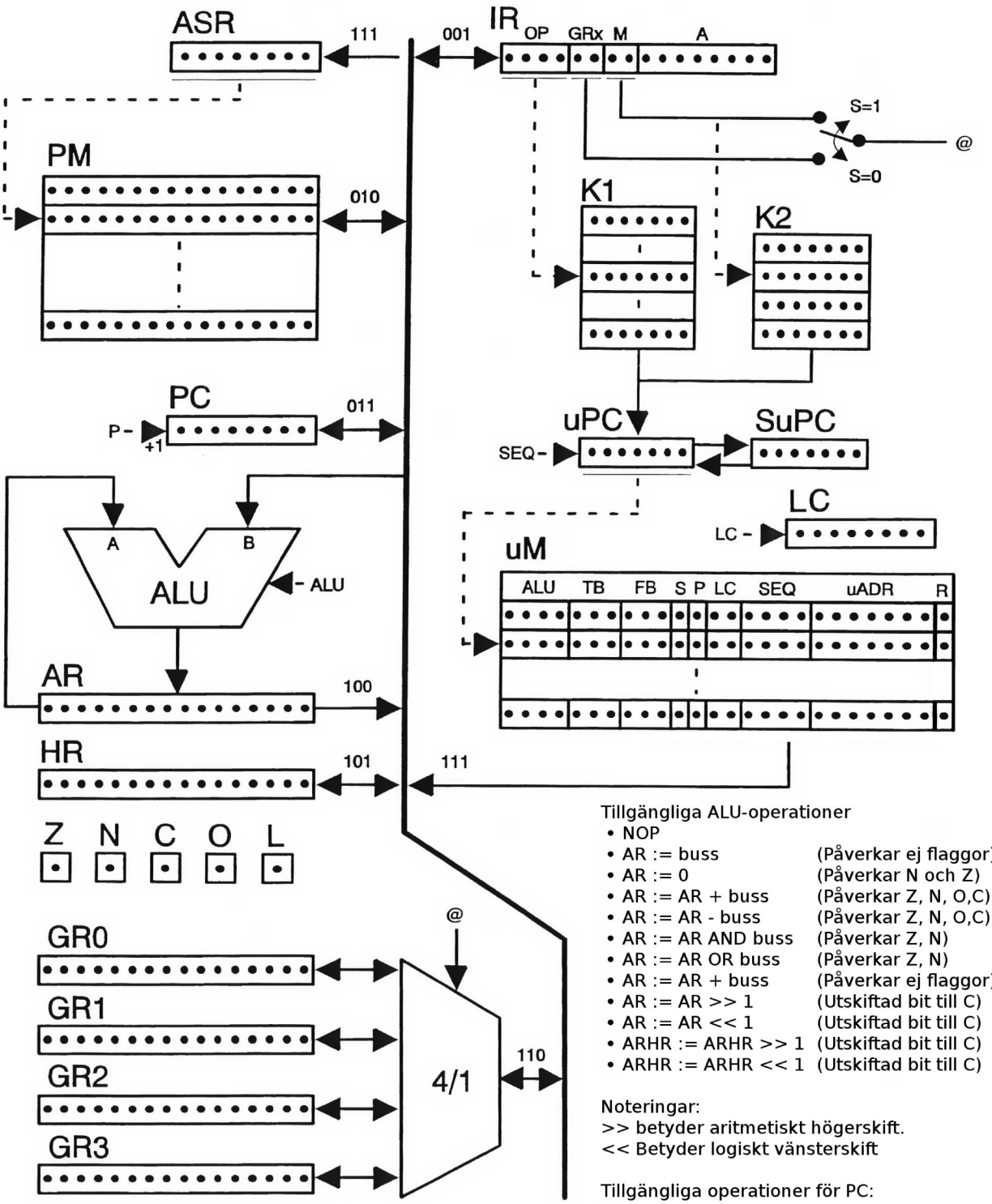
Exempel: GR0 innehåller värdet 42.

PM innehåller följande (alla värden är decimala)

Address	Värde
40	0
41	32
42	72
43	101
44	106
45	33
46	0
47	0
48	2

Efter det att instruktionen STRLEN GR0 har körts ska GR0 innehålla värdet 5.

(c) (1p) Vad händer vid exekvering av instruktionen om det inte finns något address i PM som innehåller värdet 0?



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) Vilken av följande två processorer som kan exekvera högst antal instruktioner per sekund: En 4-stegs RISC processor klockad i 2 GHz; eller en 4-vägs superskalär processor klockad i 1 GHz. Motivera ditt svar.

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

(c) (2p) Förklara vad datakonflikt (data hazard) i en RISC innebär och hur den kan lösas m h a forwarding.

(d) (2p) Vad är skillnaden mellan SRAM och DRAM? När används de olika typerna?

(f) (2p) Ange två skäl till att en MMU (Memory Management Unit) behövs i en modern dator?

Fråga 3: Binär aritmetik (7p)

(a) (1p) I ett matematikt program används 2-komplements talrepresentation för en variabel. Dess värden kan variera mellan -53 och +227. Räcker det med 8 bitar för att lagra denna variabel? Motivera ditt svar.

(b) (3p) Utrycket Y=A/2+B skall beräknas, där A är ett 8 bitars tal och både Y och B är 12 bitars tal. Både A och B använder 2-komplements talrepresentation. A = 11010100_{2C} och B = 001101011000_{2C}. Beräkna A/2 samt Y.

(c) (2p) Hur många bitar måste användas i resultatet för att kunna representera produkten av två 5-bitars tal. Antag faktorer och produkt alla använder två-komplements talrepresentation.

(d) (1p) Behöver man ta hänsyn vid beräkningen av summan av en addition att den görs av två tal i tvåkomplementsform istället för två positiva heltal?

Fråga 4: Stack och avbrott (8p)

(a) (4p) Skriv en avbrottsrutin för tutorsystemet som vid varje avbrott sätter värdet på bit 4 till värdet som läses från bit 2 (räknat med bit 0 längst till höger) i registret på adress $10080. Övriga bitar i registret får inte påverkas.

(b) (2p) Linus har i sin subrutin börjat med att i tur och ordning placera D0, D1, D2 och D3 (32 bitar var) på stacken. Nu visar det sig senare i subrutinen att han behöver hämta värdet som fanns i D1, och han löser det med kodraderna

```
ADD.L #8,A7
MOVE.L (A7),D1
SUB.L #8,A7
```

som hämtar förväntat värde tillbaks till D1. I slutet av subrutinen hämtas orginalvärdena tillbaks till D3, D2, D1 och D0. När subrutinen körs upptäcker Linus att D2 och D3 ibland inte får rätt värde när subrutinen avslutas. Varför får Linus ibland fel värde i D2 och D3 när subrutinen är slut?

(c) (2p) Vad är det för skillnad på funktionen hos instruktionerna RTE och RTS?

Fråga 5: Assemblerprogrammering (10p)

En länkad lista med värden ligger lagrad i minnet. Varje element i listan består av ett 32-bitars värde samt en 32-bitars adress till nästa element. Sista elementet i listan anger nästa adress till 0. Skriv en subrutin som lägger ihop alla värden i listan. Adressen till första elementet i listan finns i A0 vid anropet och summan ska returneras i D0. Inga register förutom D0 får vara ändrade när subrutinen returnerar.

Exempel: A0 innehåller värdet $4044 (hexadecimalt). Minnet innehåller följande (alla värden hexadecimala)

Address	Värde
$4040	$0000
$4044	$0010
$4048	$404C
$404A	$4244
$404C	$0020
$4050	$4070
$4054	$1234
$4058	$5678
$405C	$9ABC
$4060	$0100
$4064	$0000
$4068	$1010
$406C	$2020
$4070	$1000
$4074	$4060
$4078	$0100
$407C	$0000

När subrutinen returnerar ska D0 innehålla $1130 ($0010+$0020+$1000+$0100=$1130).

Fråga 6: Cache (7p)

En 4-vägs gruppassociativ datacache består av 65536 (2^{16}) bytes. En cacheline består av 32 bytes.

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

(b) (3p) Anta att cachen är tom (flushed) när k olika 32-bitars värden på adresserna n, n+4096, n+2*4096, n+3*4096 till och med n+(k-1)*4096 ska läsas om och om igen (4096 = $1000). Hur stor får k maximalt vara så att endast k cachemissar sker oberoende av hur många gånger de k olika värdena läses?

(c) (2p) En subrutin läser 16 byte från minnet på address n till och med n+15 där n är ett jämnt tal som pekar på en slumpmässig address i minnet. När profilering görs finner man att subrutinen i hälften av anropen tar tiden t_1 och i andra hälften av anropen tar tiden t_2 att utföra, där t_2 är nästan dubbelt så lång som t_1. Förklara varför detta händer.

Kort repetition av M68000

Mnemonic	Kort förklaring	Mnemonic	Kort förklaring
ADD	Addition	EXG	Exchange
ADDX	Add with X flag	EXT	Sign extend
AND	Logic and	EXTB	Sign extend a byte to 32 bit
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	OR	Logic OR
BSR	Branch to subroutine	ROL	Rotate left
CLR	Clear	ROR	Rotate right
CMP	Compare (Destination - Source)	RTE	Return from exception
DIVS	Signed division	RTS	Return from subroutine
DIVU	Unsigned division	SUB	Subtract
EOR	Logic XOR	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
```