Tentamen

Datorteknik Y, TSEA28

Datum	2016-05-31
Lokal	Kåra, T1, T2, U1, U15
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 17 Juni 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 (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).

Antag hämtfasen implementerats på samma sätt som i lab4, dvs PC pekar på instruktionen efter nuvarande instruktion när opcode avkodas.

Datorn har just nu följande instruktioner implementerade:

Opcode	Instruktion	Betydelse	Adresserings moder (M)	Påverkar flaggor
0101	JNC ADR	PC := ADR om C = 0 annars PC := PC+1	- (ange 00)	-
1010	ADD GRx,M,ADR	GRx := GRx + PM(A)	00,01,10	-
0111	STORE GRx,M,ADR	PM(A) := GRx	00,10	-

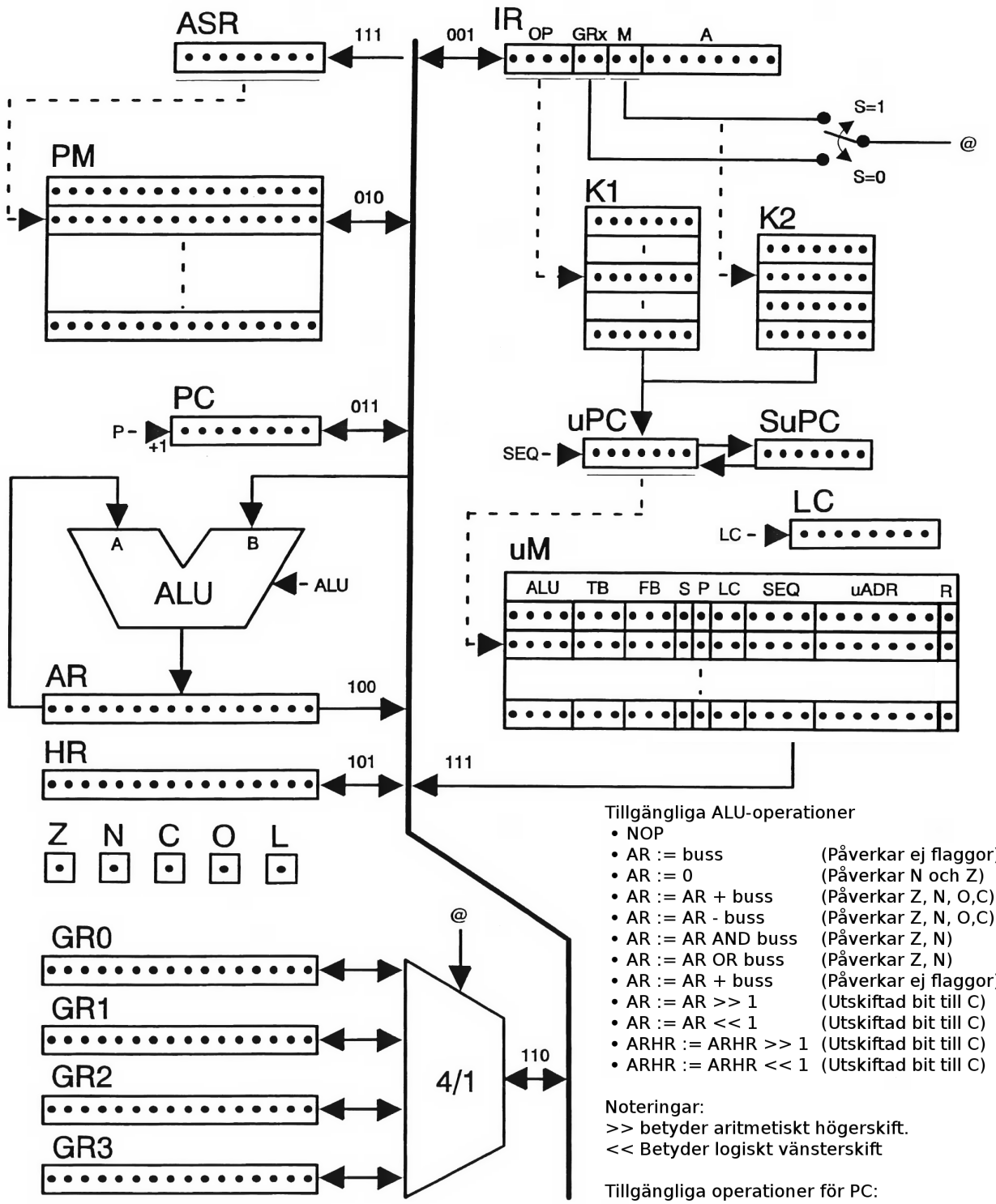
(a) (2p) Ange innehåll i minnet K1 i binär form. Ange ej definierade bitvärden som -.

(b) (7p) Implementera operationen JSR (jump subroutine). Register GR3 ska användas som stackpekare, och stacken ska växa på samma sätt som för tutor (GR3 pekar på senaste värdet lagt på stacken, nya värden som läggs på stacken hamnar på närmast lägre adress i minnet).

Opcode	Instruktion	Betydelse	Adresseringsmoder	Påverkar flaggor
0100	JSR ADR	GR3:=GR3-1, PM(GR3) := PC, PC:=ADR	- (ange 00)	-

Exempel: Register GR3 innehåller värdet $38. På adressen $10 ligger instruktionen JSR $20. När denna instruktion utförts ska registret PC innehålla värdet $20, minnesadress $37 innehålla värdet $11, samt register GR3 ska innehålla värdet $37. Flaggorna ska inte påverkas av instruktionen.

(c) (1p) Ange bitmönstret för maskininstruktionen JSR $12.



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 ett FLASH minne och ett EEPROM minne?

(b) (2p) Varför minskar prestandan för en superskalär processor när en styrkonflikt uppstår. Ge ett exempel på hur en styrkonflikt kan undvikas.

(c) (2p) Hur kan bakgrundsprogrammet i en 68000-baserad dator (tutor) förhindra att avbrott sker när en känslig del av koden körs?

(d) (2p) Ange två fördelar med att använda relativa (branch) hopp istället för absoluta (jump) hopp.

(e) (2p) Måste ett subrutinanrop använda stacken? Om inte, hur kan återhoppsadressen hanteras istället?

Fråga 3: Assemblerprogrammering (8p)

I minnet ligger en lista av adresser lagrad (varje adress är 32 bitar lång). Efter sista adressen i listan ligger adressen 0 (också 32 bitar). För varje adress i listan finns i minnet ett 32-bitars tal. Skriv en subrutin som lägger ihop alla tal som listan med adresser pekar på. Adressen till listans start finns i A0 när subrutinen anropas, och D0 ska innehålla summan.

Exempel: A0 = $0000402C, minnet innehåller

Adress	Värde	Adress	Värde
$00004020	$123B5678	$00004040	$01012020
$00004024	$11A12222	$00004044	$12121212
$00004028	$23221122	$00004048	$45642312
$0000402C	$00004024	$0000404C	$A2334001
$00004030	$00004040	$00004050	$33334444
$00004034	$00004050	$00004054	$23404422
$00004038	$00000000	$00004058	$23666022
$0000403C	$12345123	$0000405C	$A523463D

Efter anrop till subrutinen ska D0 innehålla $11A12222+$01012020+$33334444 = $45D58686.

Fråga 4: Aritmetik (6p)

(a) (2p) Antag operationen ADD.B D0,D1 beräknas då D0 = $91 och D1 = $88 beräknas (addition av två 8-bitars tal). Ange vad beräkningen ger för resultat (i binär form) och resulterande värden hos flaggorna C, V (ibland kallad O), Z och N.

(b) (2p) Antag operationen D0-D1 beräknas på registren D0 och D1, och hopp ska göras om värdet i D1 är större än eller lika med värdet i D0. Vilka värden ska flaggorna ha för att hoppet ska tas?

(c) (2p) Varför finns det två olika multiplikationsinstruktioner, en för heltal respektive en för tvåkomplementsform? Vilka skillnader finns i beräkningen av produkten för dessa två fall?

Fråga 5: Avbrott (8p)

I ett försök att ta reda på vilken del av ett program som tar mest tid på en 68000 ska avbrott skapade av en timer samla information om var i programmet processorn befinner sig. Denna avbrottsrutin ska vid varje avbrott först kontrollera om värdet på adress $5004 är 0 och i så fall bara återvända till huvudprogrammet. Om värdet inte är 0 sparas programadressen som skulle utföras när avbrottet inträffade i en buffert i minnet.

Bufferten startar på adress $4000 och lagrar 256 adresser (bufferstorlek = 4*256=1024 byte). För att hålla reda på var nästa adress ska lagras ska minnescell $5000 innehålla adressen till nästa lediga plats i bufferten. När bufferten fyllts ska avbrottsrutinen bara återvända till huvudprogrammet utan att ändra minnesadress $5000 och buffertens innehåll. Antag att minnet på adress $5000 redan innehåller värdet $4000 innan avbrottsrutinen startas första gången.

Skriv avbrottsrutinen. Timern ger avbrott på den lägsta nivån (nivå 1) och avbrott med högre prioritet ska kunna ske medan denna avbrottsrutin körs.

Exempel: Om $5000=$00004020, $5004=1, och huvudprogrammet ska starta en instruktion på adress $00001234 när avbrottet sker, så ska minnesadress $4020 innehålla $00001234, och minnesadress $5000=$00004024 när avbrottsrutinen är färdig.

Fråga 6: Cache, programförståelse (8p)

En 68000-baserad dator (med 32-bitars adressbuss) har fått en cache placerad mellan processorn och minnet. Cacheminnet är 1024 byte stort, och är direktmappad (1-vägs). Varje cacheline består av 8 byte. Både data och instruktioner lagras i denna cache.

Programmet nedan körs. Antag varje instruktion tar lika lång tid som det tar att läsa/skriva till cache/minne. Under 1 läsning kan maximalt 32 bitar läsas. Om instruktionen består av 48 bit dvs 6 bytes behövs därför 2 läsningar. Både data och instruktionsdata kan inte läsas/skrivas samtidigt. Processorn är varken pipelinad eller superskalär. Cacheträff tar 1 klockcykel, cachemiss tar 10 klockcykler.

```
ADRESS  MASKINKOD      ASSEMBLERKOD
-----------------------------------------------
1000    4280                    clr.l  D0
1002    207C 0000               move.l #$2000,A0
        2000
1008    7204                    move.l #3,D1
100a    D090           loop:    add.l  (A0),D0
100c    5088                    add.l  #4,A0
100e    5381                    sub.l  #1,D1
1010    66F8                    bne    loop
1012    4E75                    rts
```

(a) (2p) Hur många bitar används för tag-fältet för varje cacheline.

(b) (6p) Hur många klockcykler tar subrutinen att exekvera (indikera vilka adresser som adresseras och vilka som ger cachemissar)?

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
```