Tentamen

Datorteknik Y, TSEA28

Datum	2015-10-20
Lokal	TERE, TER2
Tid	8-12
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 9 och 11
Kursadministratör	Gunnel Hässler
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Onsdag 4 November 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 (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).

Datorn har just nu följande instruktioner implementerade:

Opcode	Instruktion	Betydelse	Adresserings moder (M)	Påverkar flaggor
0001	JNE ADR	PC := ADR om Z = 0 annars PC := PC+1	- (ange 00)	-
0011	LOAD GRx,M,ADR	GRx := PM(A)	00,01,10	-
0100	STORE GRx,M,ADR	PM(A) := GRx	00,10	-

Mikrokodsminnet innehåller (uPC=uPC+1 och R:=0 antas om inget annat anges).

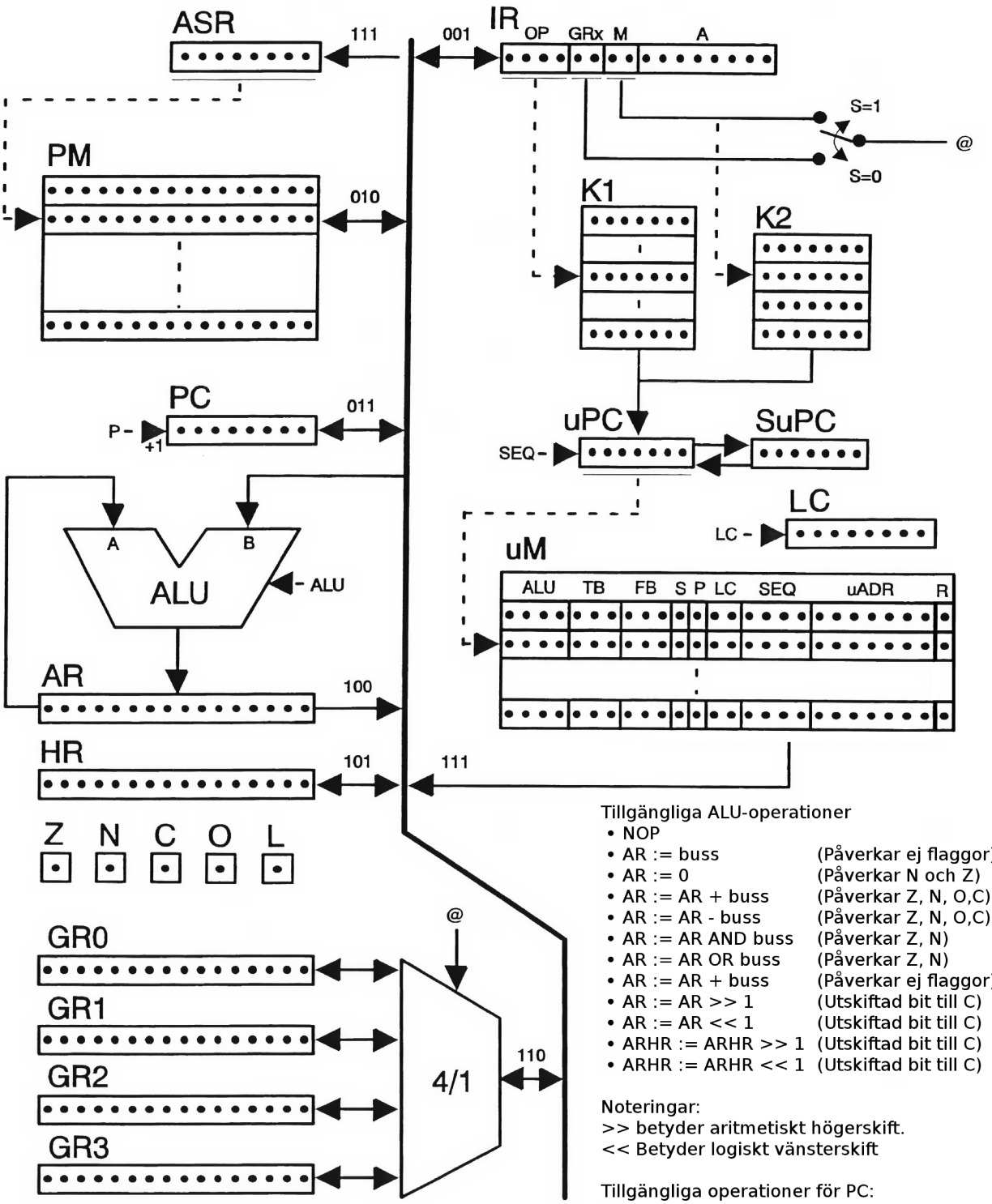
addr	Mikrokod	Kommentar
0	ASR := PC	
1	IR := PM, PC := PC+1	
2	uPC := K2(M)	
3	ASR := IR, uPC := K1(OP)	; direktadressering (M=00)
4	ASR := PC, PC := PC+1, uPC := K1(OP)	; omedelbar adressering (M=01)
5	ASR := IR	
6	ASR := PM, uPC := K1(OP)	; indirekt adressering (M=10)
7	PM := GRx, S := 0, uPC := 0	; STORE
8	uPC := 0 om Z = 1, annars uPC++	; JNE addr
9	PC := IR, uPC := 0	
10	GRx := PM, S=0, uPC := 0	; LOAD

(a) (7p) Skriv mikrokod för instruktionen CMPSWP GRx,M,ADR. Instruktionen ska jämföra innehållet i GRx med innehållet i PM. Om värdet i GRx är mindre än innehållet i PM ska värdet i GRx byta plats med värdet i PM. l. Flaggorna ska efter instruktionen indikera om bytet skett eller inte med C-flaggan (C=1 om GRx < PM). Z=0 om Grx=PM, O,N odefinierade. Antag värden i register/minne är positiva heltal.

Opcode	Instruktion	Betydelse	Adresserings-moder	Påverkar flaggor
0111	CMPSWP GRx,M,ADR	Byt plats GRx och PM(A) om GRx < PM(A)	00,10	Z,N,O,C

Exempel: GR2 = 22, PM(3) = 33. Efter CMPSWP GR2,00,3 ska GR2 = 33, PM(3) = 22, Z=0, C=1.

(b) (2p) Ange innehåll i minnet K1



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) Ange två typer av ROM.

(b) (2p) Vad är utmärkande för en SIMD-processor?

(c) (2p) Ange två fördelar med avancerade bussar som PCI-express jämfört med bussen i tutor-systemet.

(d) (2p) Vad är skillnaden mellan big-endian och little-endian processorer?

(e) (2p) Finns det andra sätt än assemblerinstruktioner som påverkar stackens innehåll på MC68000, dvs påverkas stacken av annat än t ex JSR subrutin, MOV D0,-(A7) etc.

Fråga 3: Avbrott (10p)

32 bitar seriellt data ska skickas ut via bit 0 på PIA kretsens port A på tutorsystemet. En klocka är ansluten till PIA:ns port A:s avbrottsingång. Det seriella datat som ska skickas ut ska vara placerat på adress $3000-$3003, och adressen $3004 anger om nytt data finns som ska skickas. $3004 = 0 betyder inget data att skicka och $3004=1 att data håller på att skickas. Adresserna $3010-3020 är också reserverade för användning av avbrottsrutinen och får användas i avbrottet. Data ska skickas med LSB först. När alla 32 bitar skickats ut ska värdet på adress $3004 sättas till 0 och bit 0 på port A sättas till 0.

De andra bitarna i port A ska inte påverkas av avbrottsrutinen.

Följande förväntas ske: Huvudprogrammet skriver ett 32-bitars värde till adress $3000 och därefter sätter $3004 = 1. Avbrottsrutinen ska vid nästa avbrott skicka ut LSB-biten (bit 0) på port A:s bit 0. Vid avbrottet därefter ska bit 1 skickas ut på port A:s bit 0. När avbrottet körs för 32:a gången ska MSB-biten skickas ut på port A:s bit 0. Vid avbrottet efter detta ska bit 0 på port A sättas till 0 och värdet på adress $3004 sättas till 0. Avbrotten därefter ska inte påverka bit 0 på port A förrän $3004 på nytt sätts till 1 och ytterligare 32 bitar då skickas.

Skriv denna avbrottsrutinen.

Exempel: Antag $3000 sätts till $92345672 och $3004 sätts till 1. De efterföljande avbrotten ska då sätta bit 0 på port A till värdena 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, ... 1, 0, 0, 1. Efter dessa 32 avbrott ska nästa avbrott sätta bit 0 på port A till 0 och $3004 sätts till 0 av avbrottsrutinen.

Fråga 4: Assemblerprogrammering (8p)

Skriv en subrutin som skapar en vektor med fibonaccis talföljd. Register A0 anger adressen där vektorn startar, D0 anger hur många värden som ska sparas. Varje värde ska lagras som ett 16-bitars tal. Inga register får vara ändrade när subrutinen är klar.

Fibonaccis talföljd är 0, 1, 1, 2, 3, 5, 8, 13, 21 etc. Dvs F(0) = 0, F(1) = 1, och F(n) = F(n-1) + F(n-2) för n>1.

Exempel: D0 = $04, A0= $4000 => $4000=$0000, $4002=$0001, $4004=$0001, $4006=$0002

Fråga 5: Aritmetik (6p)

(a) (2p) Vad är det minsta antal bitar som behövs för att representera talet 7 i 2-komplementsform?

(b) (2p) 8 stycken N-bitars 2-komplementstal ska adderas. Hur många bitar måste minst användas för korrekt kunna representera summan?

(c) (2) Följande programdel hittas i en rutin. Vad beräknar rutinen för värde på D0? Antag beloppet av D0 < 1024.

```
MOVE.W D1,-(A7)     ; Spara D1 på stack
MOVE.W D0,D1
ASL.W #2,D0         ; Skifta D0 två bitar åt vänster
ADD.W D1,D0
MOVE.W (A7)+,D1     ; Återställ D1
```

Fråga 6: Cache (7p)

En processor har en så kallad level 1 cache på 4 Kbyte (4096 bytes). Denna cache är direktmappad och varje cacheline består av 16 byte. Adressbussen hos processorn består av 32 bitar.

(a) (2p) Hur många olika bitar används för att representera tag-värdet i cachen?

(b) (1p) Hur många jämförelser görs mellan adressens tag-del och lagrade tag i cachen vid varje minnesaccess?

(c) (2p) En matris ligger lagrad i minnet och adresserna läses i sekvensen $4000, $4100, $4200, $4300 etc. Antag cacheminnet från början är tomt. Hur många adresser kan läsas i sekvensen innan data från den första läsningen på adress $4000 tas bort ur cachen?

(d) (2p) Antag cachelines storlek ökas till 32 byte. Hur påverkas tiden att läsa från minnet om man får en cachemiss? Antag primärminnet består av DRAM.

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
```