Tentamen

Datorteknik Y, TSEA28

Datum	2017-06-02
Lokal	G35, 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	Måndag 19 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! Läs anvisningarna på tentaomslaget!

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).

Mikrokodsminnnet innehåller från början

addr	Mikrokod	Kommentar
0	ASR := PC, uPC := uPC+1	
1	IR := PM, PC := PC+1, uPC := uPC+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, uPC := uPC+1	
6	ASR := PM, uPC := K1(OP)	; indirekt adressering (M=10)
7	PM := GRx, S := 0, R := 0, uPC := 0	; STORE
8	uPC := 0 om Z = 1, annars uPC++	; JNE addr
9	PC := IR, uPC := 0	
10	GRx := PM, S := 0, R := 0, uPC := 0	; LOAD

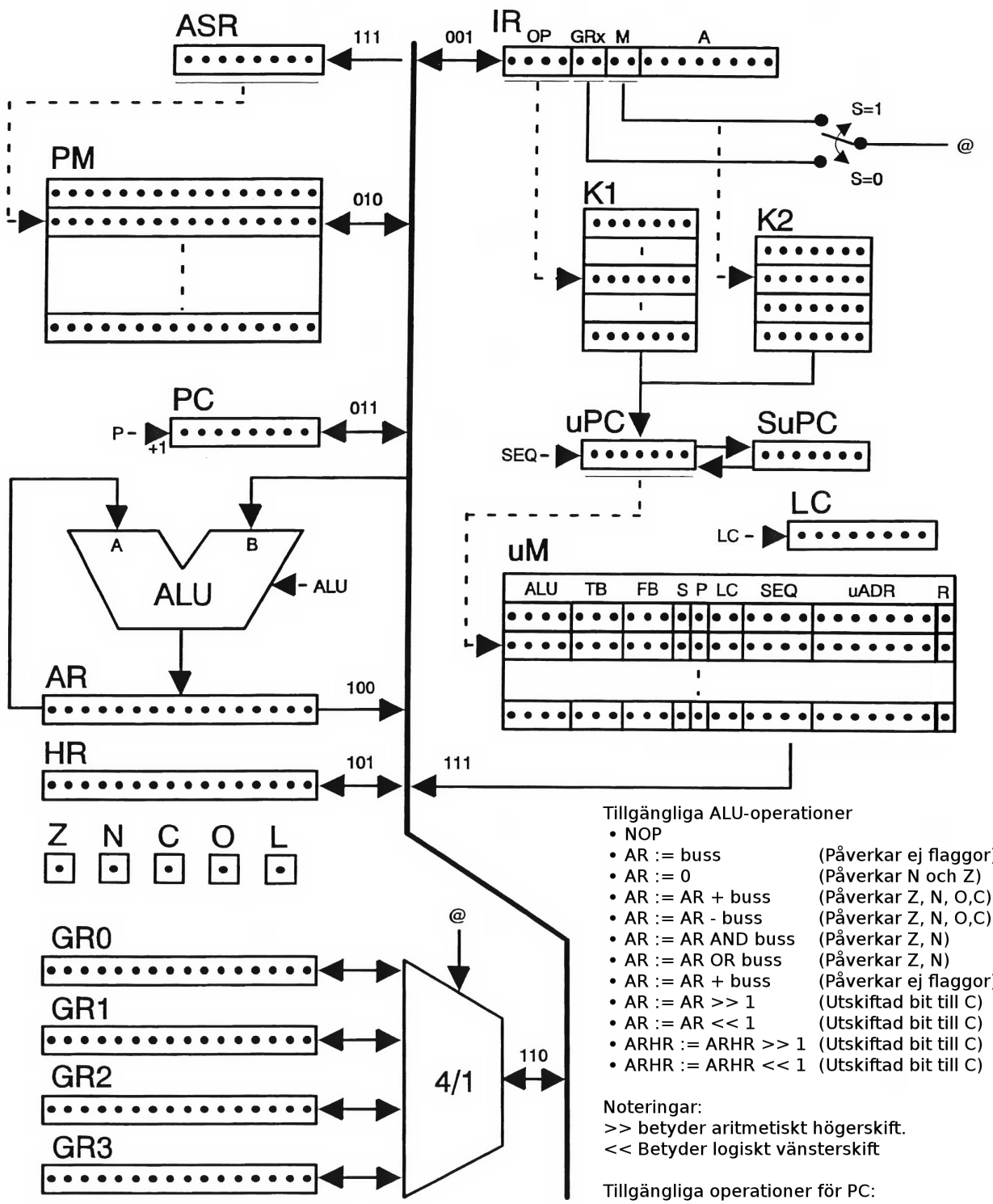
Instruktionsuppsättningen ska i en framtida version utökas med instruktioner som arbetar med 32-bitar data. Därför behöver mikrokoden för addresseringsmoderna justeras så en kopia av värdet som finns i ASR också sparas i HR.

(a) (2p) Ändra/utöka mikrokoden så HR innehåller en kopia av värdet i ASR efter att mikrokod tillhörande adresseringsmoderna M=00, M=01 och M=02 har utförst.

(b) (2p) Ange innehållet i K2 i binär form efter ändringen i deluppgift a) ovan. Ange '-' om ett värde inte är definierat.

(c) (6p) Implementera instruktionen ADDNZ GRx, GRy. M-fältet anger GRy. Om Z=0 så ska den beräkna GRy = GRy+GRx och resultatet av den additionen ska påverka flaggorna. Om Z = 1 så ska inte additionen utföras, och flaggorna ska då inte heller påverkas.

Exempel: GR1 = $1234, GR2 = $7621, Z=0. Efter ADDNZ GR1,GR2 är nya värdet i GR2 = $8855 och Z = 0, O = 1, N = 1 samt C = 0. **Exempel:** GR3 = $1224, GR1 = $5621, Z=0. ADDNZ GR3,GR1 påverkar inte GR1, GR3 och flaggorna.



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) Motivera varför branch prediction är en viktig del av en pipelinad RISC-processor.

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

(c) (2p) Vad är skillnaden mellan SRAM och DRAM minnen?

(d) (2p) Vad är DMA, och vad kan det användas till?

(e) (2p) Varför behövs cacheminne i en modern processor när det inte behövdes i äldre processorer (t ex hos en 68000 processor i TUTOR-systemet)?

Fråga 3: Assemblerprogrammering (10p)

En krypteringssubrutin ska skrivas. Den ska kryptera ett meddelande lagrat i minnet med hjälp av ett 8-bitars värde (nyckeln) och en enkel kryptering baserad på bitvis XOR. Varje byte i meddelandet ska XOR:as med nyckelns värde, och efter varje XOR ska nyckelns värde räknas upp med 1. Sista byte i meddelandet har värdet $00, och denna byte ska inte krypteras. Register A0 ska innehålla adressen där meddelandet börjar och D0 innehåller nyckeln som ska användas på 1:a byten i meddelandet.

Exempel: A0 = $00004022, D0 = $52, och minnet innehåller

Adress	Värde	Adress	Värde
$00004020	$123B5678	$00004028	$005123B0
$00004024	$11A12222	$0000402C	$CD121212

Efter anrop till krypteringssubrutinen får A0 och D0 innehålla valfria värden, medan minnet ska innehålla

Adress	Värde	Adress	Värde
$00004020	$123B042B	$00004028	$005123B0
$00004024	$45F47475	$0000402C	$CD121212

Dvs beräkna och spara $56 XOR $52, $78 XOR $53, $11 XOR $54, $A1 XOR $55 etc. Avsluta subrutinen när värdet $00 hittas på adress $4028.

Fråga 4: Aritmetik (6p)

(a) (2p) Ange summan av de två 16-bitars 2-komplementstalen $B203 + $FFAD på binär form.

(b) (2p) Vilka värden får flaggorna Z,C,N och O (overflow) vid beräkning av $B203 + $FFAD i en 16-bitars processor?

(c) (2p) Beräkna summan av det 4-bitars 2-komplementstalet 1010 med ett 5-bitars 2-komplementtal 11011. Beskriv summan som ett 6-bitars 2-komplementtal.

Fråga 5: Avbrott (8p)

En fläktkontroller till en dator skall implementeras. Denna ska styras med hjälp av en avbrottsrutin som startas var 10:e ms. På adress $10080 anger bit 5 till och med bit 0 (ett 6-bitars positivt heltal) den aktuella temperaturen. Bit 6 och bit 7 är odefinierade. Avbrottsbegäran nollställs när adressen $10080 läses. På adress $10084 används bit 4 till och med bit 0 för att styra fläktens hastighet (ett 5-bitars positivt heltal, högre värde motsvarar högre varvtal på fläkten), och om bit 7 på adress $10084 sätts till 1 så minskas datorns klockfrekvens. Ett målvärde för temperaturen finns lagrat på adress $5000 i minnet.

Fläktkontrollern ska om temperaturen är lika med målvärdet inte ändra värdet på adress $10084. Om temperaturen är lägre än målvärdet ska varvtalet minskas med 1 (minsta tillåtna värde är 0) och bit 7 på port $10084 nollställas till 0. Om temperaturen är högre än målvärdet och varvtalet inte har sitt maximala värde så ska varvtalet på fläkten ökas med 1. Om temperaturen är högre än målvärdet och varvtalet redan har sitt maximala värde så ska istället datorns klockfrekvens minskas genom att sätta bit 7 på adress $10084 till 1. Antag att värdet på port $10084 initierats till $00 innan 1:a avbrottet kommer.

Exempel: $5000 = $04, $10080 = $42, $10084 = $92. Avbrottsrutinen ska då sätta $10084 = $11.

Exempel: $5000 = $10, $10080 = $21, $10084 = $1F. Avbrottsrutinen ska då sätta $10084 = $9F.

Fråga 6: Cache (6p)

Den senaste ARM-processorn Cortex A73 har en nivå1 (L1) datacache som är 64 KByte stor (1 KB = 1024 byte). Denna cache är 16-vägs gruppassociativ där varje cacheline är 64 byte. Processorn använder 64-bitar lång adresser.

Ett program ska läsa en sekvens av bytes med start på en slumpmässig adress n. Antag cachen är tom när denna läsning startas.

(a) (2p) Hur många cachelines finns i cacheminnet totalt?

(b) (2p) Hur många bitar av adressen används som index i cacheminnet?

(c) (2p) Hur många byte av sekvensen kan i medeltal läsas efter den första cachemissen på adress n innan en ny cachemiss fås.

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
```