Tentamen

Datorteknik Y, TSEA28

Datum	2024-08-20
Lokal	TER4, TERE
Tid	14-18
Utb. kod	TSEA28
Modul	DIT1
Kursnamn *Provnamn*	Datorteknik Y Digital tentamen
Institution	ISY
Antal frågor	6
Antal sidor (inklusive denna sida)	6
Kursansvarig	Kent Palmkvist, 1347, Kent.Palmkvist@liu.se
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Kent Palmkvist 013-281347
Besöker skrivsalen	Ca kl 15 och kl 17 (+/- 30 minuter)
Kursadministratör	Ulrika Ericsson, 013 282379
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5

Viktig information

- Hexadecimala värden anges antingen som 0x1234 eller $1234
- 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 du 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).

Mikrokodsminnnet innehåller i denna datormodell bland annat

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, uPC := uPC + 1	; omedelbar adressering (M=01)
5	PC := PC + 1, uPC := K1(OP)	
6	ASR := IR, uPC := uPC+1	; indirekt adressering (M=10)
7	ASR := PM, uPC := K1(OP)	
8	AR := IR, uPC := uPC+1	; indexerad adressering (M=11)
9	AR := GR3+AR, uPC := uPC+1, R:=1	
0xa	ASR := AR, uPC := K1(OP)	

(a) (6p) Skriv mikrokod för instruktionen PUSH GRx (med M=10 och A-fält = 0x01). Instruktionen ska placera värdet i register GRx på en stack. Adressen till första lediga platsen på stacken lagras i GR2. Stacken växer mot lägre adresser. De mest signfikanta 8 bitarna i GR2 kan ignoreras (dvs får påverkas av PUSH och behöver inte tas hänsyn till).

Exempel: Antag GR1=0xABCD, GR2=0x0034, PM(0x34)=0x1234. Om instruktionen PUSH GR1, M=01, A=0x01 körs resulterar detta i GR2=0x0033, PM(0x34)=0xABCD.

(b) (2p) Ange innehållet i minnet K1 i binär form. Antag att förutom PUSH (opcode 9) så finns även instruktionerna LOAD (opcode 5, startadress 0x30), STORE (opcode 2, startadress 0x45), och ADD (opcode 0xB, startadress 0x50). Ange både adress och data i K1 i binär form, och ej definierade instruktioner ska starta på adress 0.

(c) (2p) Ange objektkoden (minnesinnehåll i PM) för instruktionen PUSH GR3,M=01,A=0x01. Ange svaret i hexadecimal form.

Fråga 2: Allmän teori (10p)

(a) (2p) Vad är talområdet för ett 9-bitars 2-komplementstal respektive ett 9-bitars positivt heltal?

(b) (2p) Varför måste alla datorer innehålla icke-volatilt minne?

(c) (2p) Beskriv hur pipelining fungerar och motivera varför det ökar prestandan hos en RISC-processor.

(d) (2p) Motivera varför/hur virtuellt minne kan garantera att två olika program inte kommer åt varandras data i minnet.

(e) (2p) Vad är skillnaden mellan EEPROM och ROM?

Fråga 3: Assemblerprogrammering (10p)

Skriv en subrutin som avkodar en RLL-kodad datamängd och skickar den till en annan subrutin kallad Subrutin1. Indata är lagrat i en tabell med 16-bitars värden. Varje värde består av två delar, där bit 9 t om bit 0 är ett 10-bitars 2-komplementsvärde och bit 15 t o m bit 10 är ett 6 bitars positivt heltal. Det 6-bitars positiva heltalet anger hur många kopior av det 10-bitars tvåkomplementsvärdet som ska skickas till subrutinen kallad Subrutin1. Om det 6-bitars positiva heltalet är 0 är tabellen slut och subrutinen ska returnera.

Subrutinen Subrutin1 förväntar sig indata som ett 32-bitars 2-komplementvärde i register r0. Subrutinen Subrutin1 förstör även r0 och r1. Subrutinen Subrutin1 ska inte skrivas (antag den finns definierad på annan plats).

Vid anrop till subrutinen innehåller r0 adressen till första värdet i tabellen. Minnescellerna som innehåller tabellen får inte ändras av subrutinen.

Exempel: r0 = 0x20001002,

Minnesinnehåll

Adress	Värde	Adress	Värde
0x20001000	0x1076	0x20001008	0x0234
0x20001002	0x0E00	0x2000100A	0x1945
0x20001004	0x2567	0x2000100C	0x3F71
0x20001006	0x430B	0x2000100E	0x0002

Första värdet i tabellen (0x0E00) delas upp i 6-bitars positiva heltalet 0x03 samt 10-bitars tvåkomplementstalet 0x200 (teckenförlängs till 32 bitars längd och blir då 0xFFFFFE00). Subrutin anropas därefter 3 gånger med r0=0xFFFFFE00 vid varje anrop. Därefter görs på motsvarande sätt med 0x2567 och 0x430B som medför att Subrutin1 anropas 9 gånger med r0=0x00000167 samt 4 gånger med r0=0xFFFFFF0B. Värdet efter det (0x0234) innehåller 6-bitars positiva heltalet 0x00 och subrutinen returnerar därför.

Fråga 4: Avbrott (8p)

Skriv en avbrottsrutin som först läser en byte från adress 0x40001000 (adress 0x40001001 får inte läsas). Om bit 3 = 0 och bit 5 = 1 i detta värde ska subrutinen Subrutin2 anropas. Slutligen (både ifall Subrutin2 anropats eller inte) ska adressen 0x40001010 läsas 4 gånger (adress 0x40001011 får inte läsas) och kombineras till ett 32 bitars värde (bit 7 t om bit 0 från första läsningen, bit 15 t om bit 8 från andra läsningen, etc.). Slutligen ska det 32-bitars värdet som skapats skrivas till adress 0x40001020 (som en 32-bitar skrivning).

Subrutin2 förändrar värdet i register r4. Subrutin2 ska inte skrivas, utan antas vara definierad på annan plats.

Inga andra avbrott får starta medan denna avbrottsrutin kör.

Exempel: Avbrottsrutinen läser in värdet 0x25 från adress 0x40001000. Då bit 3=0 och bit 5=1 i detta värde anropas Subrutin2. Slutligen läses adress 0x40001010 4 gånger och får värdena 0x12, 0x34, 0x56, respektive 0x78. Till slut skrivs då värdet 0x78563412 till adress 0x40001020.

Fråga 5: Aritmetik (6p)

(a) (2p) Byt tecken på det 20 bitar långa 2-komplementstalet 0x5A2CE. Svaret ska ges som ett 20-bit 2-komplementstal i hexadecimal form.

(b) (2p) Antag summan 0x7E + 0x83 beräknas i en 8-bitars dator. Ange resulterande värden på flaggorna C, Z, N, V/O.

(c) (2p) Beräkna summan $011010_{2C} + 1011_{2C}$ (båda är 2-komplementstal med 6 respektive 4 bitars ordlängd). Ange svaret som ett 10-bitars 2-komplementstal i binär form.

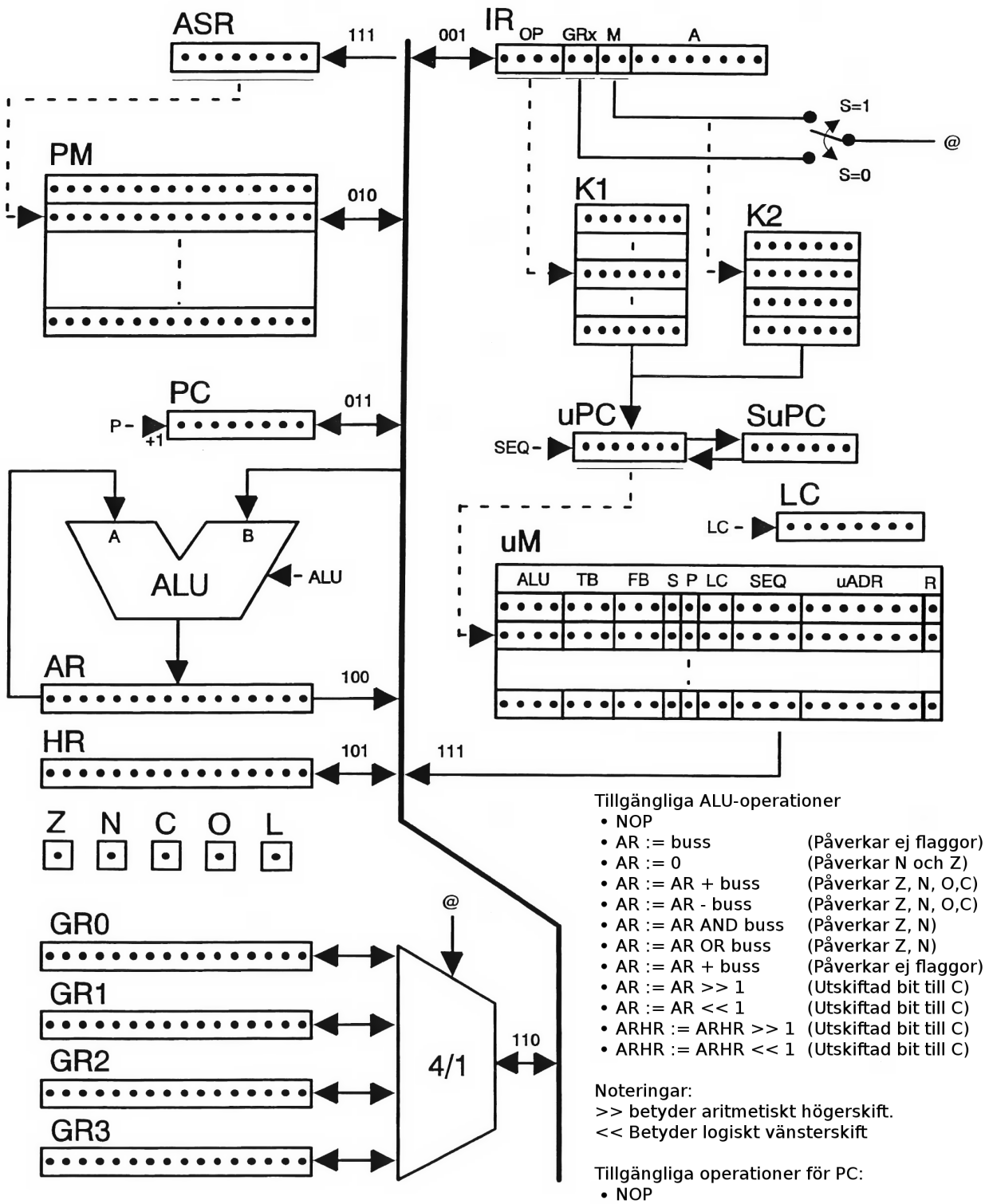
Fråga 6: Cache (6p)

En 2-vägs gruppassociativ cache är 32KB stor (32768 byte). Varje cacheline består av 32 byte. Adressbussen består av 24 bitar.

(a) (2p) Hur många bitar av adressen används som tag?

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

(c) (2p) Antag cachen är tom. Hur länge kan adresser läsas enligt sekvensen 0x123456+n*0x800 (n = 0,1,2,3,.....) innan tidigare inläst data kastas ut ur cachen?



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)

Kort repetition av ARM Cortex-M4

Mnemonic	Kort förklaring	Mnemonic	Kort förklaring
ADR	Address	CPSID	Disable interrupt
ADD, ADDS	Addition	CPSIE	Enable interrupt
ADDC,ADDCS	Addition with C flag	EOR,EORS	Logic XOR
AND,ANDS	Logic AND	LDR	Load register
ASR,ASRS	Arithmetic shift right	LDRB,LDRSB	Load register byte
B	Branch	LDRH,LDRSH	Load register halfword
BCC,BLO	Branch on carry clear	LSL,LSLS	Logic shift left
BCS,BHI	Branch on carry set	LSR,LSRS	Logic shift right
BEQ	Branch on equal	MOV,MOVS	Move
BGE	Branch on greater than or equal	MUL,MULS	Multiply
BGT	Branch on greater than	MVN	Move not
BL	Branch and link	NOP	No operation
BLT	Branch on less than	ORR,ORRS	Logic OR
BLE	Branch on less than or equal	POP	Pop
BLS	Branch on lower or same	PUSH	Push
BMI	Branch on minus	ROR	Rotate right
BNE	Branch on not equal	SBC,SBCS	Subtraction with C flag
BPL	Branch on plus	STR	Store register
BVC	Branch on overflow clear	STRB	Store register byte
BVS	Branch on overflow set	STRH	Store register halfword
BX	Branch and exchange	SUB,SUBS	Subtraction
CMP	Compare (Destination - Source)	TST	Test

```
; Exempel på ARM Cortex-M4 kod som kopierar 200 bytes från 0x2000 till 0x3000

main:  mov  r0,#(0x2000 & 0xffff)
       movt r0,#(0x2000 >> 16)
       mov  r1,#(0x3000 & 0xffff)
       movt r1,#(0x3000 >> 16)
       mov  r3,#50
loop:
       ldr  r4,[r0],#4
       str  r4,[r1],#4
       subs r3,r3,#1
       bne  loop
```