Tentamen (Exempel)

Datorteknik Y, TSEA28

Datum	2018-05-29
Lokal	KÅRA,T1,T2,
Tid	14-18
Kurskod	TSEA28
Provkod	TEN1
Kursnamn *Provnamn*	Datorteknik Y Skriftlig tentamen
Institution	ISY
Antal frågor	6
Antal sidor (inklusive denna sida)	7
Kursansvarig	Kent Palmkvist, 1347, kentp@isy.liu.se
Lärare som besöker skrivsalen *Telefon under skrivtiden*	Kent Palmkvist 1347, 013-281347
Besöker skrivsalen	Ca kl 15 och kl 17
Kursadministratör	Gunnel Hässler
Tillåtna hjälpmedel	Inga
Betygsgränser	Preliminärt 0-20: U, 21-30: 3, 31-40: 4, 41-50: 5
Visning	Onsdag 13 Juni kl 13.00 – 14.00 i kursansvarigs kontor

Viktig information

- Hexadecimala värden kan anges 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 (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).

Mikrokodsminnnet innehåller bland annat

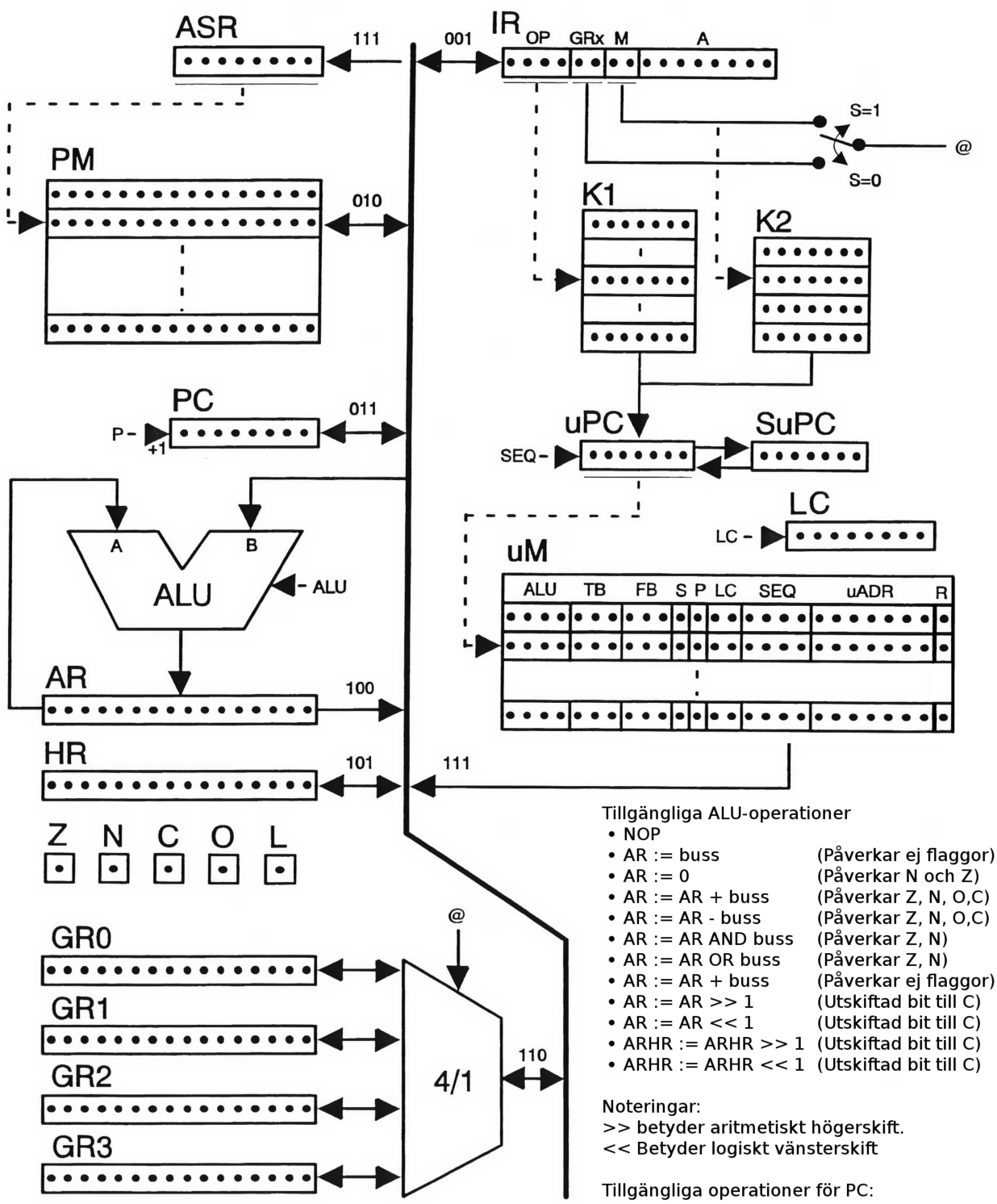
addr	Mikrokod	Kommentar
0	ASR := PC, uPC := uPC+1	
1	IR := PM, PC := PC+1, uPC := uPC+1	
2	uPC := K2(M)	
3	ASR := IR, uPC := K1(OP)	; direktadressering (M=00)
4	ASR := PC, PC := PC+1, uPC := K1(OP)	; omedelbar adressering (M=01)
5	ASR := IR, uPC := uPC+1	; indirekt adressering (M=10)
6	ASR := PM, uPC := K1(OP)	

a) (7p) Skriv mikrokod för instruktionen MAX GRx, A med adresseringsmoder 0 och 2 (direktadressering respektive indirekt adressering). Jämför värdet i GRx med värdet i minnet. Det största värdet ska placeras i GRx. Värdena ska ses som positiva heltal. Flaggornas värde får förstöras.

Ange även adresserna för mikrokodsinstruktionerna.

Exempel: GR0 = 0x1234. PM(0x12)=0x8222. Efter MAX GR0,0x12 är värdet på GR0 = 0x8222.

b) (2p) De fem instruktionerna STORE, LOAD, BEQ, AND och ADD startar på mikrokodadress 15, 17, 23, 19, respektive 27. Opcode för instruktionerna ska vara 1, 7, 2, 9, respektive 0. Beskriv innehållet i minnet K1 i binär form.



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 processortyperna pipelinad RISC respektive mikroprogrammerad CISC ökar beräkningshastigheten mest genom att branch prediction läggs till. Motivera ditt svar.

(b) (2p) Vad är DMA?

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

(d) (2p) Vad skiljer en von-Neumann arkitektur från en Harvard-arkitektur?

(e) (2p) Vad är en styrkonflikt (control hazard) i en pipelinad processor?

Fråga 3: Assemblerprogrammering (10p)

Skriv en subrutin som samlar ihop data (32-bitar långa) som ligger utspritt i minnet i form av en länkad lista och placerar dessa i en sammanhängande vektor. Efter varje 32-bitars värde finns en adress lagrad som anger var nästa värde finns i minnet. Efter det sista värdet anges adressen 0. Register r0 ska innehålla adressen till 1:a värdet, r1 anger var vektorn börjar där data ska sparas. Värdet i register r3 ska ange hur många värden som kopierats.

För 68000 använd A0 istället för r0, A1 istället för r1 och D0 istället för r3

Exempel: r0 = 0x20001034, r1 = 0x0x20002000

Adress	Värde	Adress	Värde
0x20001000	0x11112222	0x20001020	0x76541234
0x20001004	0x33334444	0x20001024	0x20001008
0x20001008	0x17263540	0x20001028	0xababcdcd
0x2000100c	0x00000000	0x2000102c	0x23324444
0x20001010	0xabcd0123	0x20001030	0x20406070
0x20001014	0x20001020	0x20001034	0x12345678
0x20001018	0x55556666	0x20001038	0x20001010
0x2000101c	0x77778888	0x2000103c	0x88884848

När subrutinen är klar ska minnet på adress $20002000 se ut som nedan och register r3 = 0x00000004.

Adress	Värde
0x20002000	0x12345678
0x20002004	0xabcd0123
0x20002008	0x76541234
0x2000200c	0x17263540

Fråga 4: Avbrott (8p)

En avbrottrutin ska skrivas, som läser av 4 olika indata (1 byte stora var, sedda som positiva heltal) från adresserna 0x40001000, 0x40001010, 0x40001020, och 0x40001030. Endast de värden som är udda (minst signifikant bit i datat = 0) ska sedan jämföras. Det största av dessa udda värden ska sedan skrivas till adress 0x40001040. Om inget udda värde lästs så ska värdet 0x00 skrivas på adress 0x40001040. Inga andra adresser i adressområdet 0x40000000-0x40002000 får läsas/skrivas.

Exempel: Om värden som läses från adresserna 0x40001000 t o m 0x40001030 är 0x88, 0x33, 0x41 respektive 0x11, så ska 0x41 skrivas till adress 0x40001040.

Fråga 5: Aritmetik (7p)

(a) (2p) Skriv om det 3-bitars 2-komplementstalet ”101” som ett 7-bitars tal i 2-komplementsform.

(b) (2p) Beräkna differensen av 0x1b-0x2d om subtraktionen beräknas mha omskrivningen av subtraktionen till en addition, dvs x-y = x+(-y). Skriv först upp upp additionen i binär form innan summeringen utförs. Indata och differensen är 7 bitar lång.

(d) (3p) Beräkna multiplikationen av ”1101” * ”0101” (positiva heltal). Ange svaret i binärform.

Fråga 6: Cache (6p)

Processorn i Raspberry Pi 3 har en 16 KByte (16384 bytes) Level 1 cache med en cacheline längd av 64 byte. Denna cache är en 4-vägs gruppassociativ cache. Adressbussen är 32 bitar lång.

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

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

(c) (2p) Varje instruktion i processorn är 32 bitar lång. Hur många instruktioner får plats i en cacheline?

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
```

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 negative
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
BLT	Branch on less than	ROR	Rotate right
BMI	Branch on minus	SBC,SBCS	Subtraction with C flag
BNE	Branch on not equal	STR	Store register
BPL	Branch on plus	STRB	Store register byte
BVC	Branch on overflow clear	STRH	Store register halfword
BVS	Branch on overflow set	SUB,SUBS	Subtraction
BX	Branch and exchange	TST	Test
CMP	Compare (Destination - Source)		

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

addr1: .field 0x2000,32
addr2: .field 0x3000,32

main:  ldr  r0,addr1
       ldr  r1,addr2
       mov  r3,#50
loop:
       ldr  r4,[r0],#4
       str  r4,[r1],#4
       subs r3,r3,#1
       bne  loop
```