

Omtentamen i CDT204 - Datorarkitektur

2012-11-05

Skrivtid: 08.10-12.30

Hjälpmedel: Miniräknare och valfritt skriftligt (ej digitalt) material.

Lärare: Stefan Bygde, kan nås på 070-619 52 83.

Tentamen är uppdelad i fyra delar som motsvarar de fyra inlämningsuppgifterna i kursen. Du behöver bara göra de delar som du ännu inte blivit godkänd på. Varje del betygsätts individuellt enligt följande:

- 3: 50%
- 4: 70%
- 5: 90%

Betyget för tentamentsmomentet bestäms av medelvärdet av betygen på de individuella delarna avrundat till det närmaste hela betygssteget (om det är lika långt till två olika betygssteg så avrundas detta uppåt).

Viktigt att tänka på:

- Ge så fullständiga svar som möjligt för att erhålla full poäng. Alla beräkningar, approximationer, antaganden och motiveringar ska redovisas för full poäng då inget annat anges. Använd gärna figurer och exempel för att förtydliga.
- Om du inte förstår en uppgift så kan du ringa läraren och fråga.
- Oläsliga lösningar ger inga poäng.
- Skriv tydligt både DEL och UPPGIFT på varje blad

Del I: Talsystem, prestanda och instruktionsuppsättningar (10p)

Uppgift 1: Talbaser (2p)

Det decimala talet 101 är givet.

- (a) Hur ser den binära representationen av talet ut?
- (b) Hur ser den hexadecimala (basen 16) representationen av talet ut?

Uppgift 2: Prestanda (4p)

Ett visst program kompileras för en processor som har klockfrekvensen 2,5 GHz. Den resulterande maskinkoden består av $2,3 \times 10^6$ instruktioner, och på den aktuella processorn får den en IPC (instruction per clock-cycle) på 0,8. Hur lång tid tar programmet att exekvera?

Uppgift 3: MIPS (4p)

Givet är följande MIPS-assembly-kod:

1. `sw $t0, 64($s1)`
2. `addi $t0, $t0, 7`
3. `addi $t1, $t1, -34`
4. `add $s0, $zero, $t0`
5. `beq $t1, $zero, L0`
6. `lw $s0, 64($s1)`
7. `L0:`

Om registret `t0` innan koden körs har värdet 23 och registret `t1` har värdet 34, vad kommer registret `s0` ha för värde vid programpunkten `L0`?

Del II - Pipelining (12p)

Uppgift 1 (6p)

Följande CPU-pipeline är given.

Steg	Tid (psek)
A	425
B	500
C	400
D	450

Antag att inga bubblor någonsin uppstår i pipelinen.

- (a) Vad ger pipelinen för svarstid för en enstaka instruktion?

- (b) Vad är genomströmningen i MIPS (miljoner instruktioner per sekund) för ett program bestående av x instruktioner (utan hoppinstruktioner), där $x > 0$?
- (c) Antag att man ska konstruera en processor som inte använder pipelining och som exekverar en instruktion per klockcykel. Vilken klockcykeltid måste denna processor ha för att ge samma exekveringstid som processorn ovan för ett program bestående av 10 instruktioner (utan hoppinstruktioner)?

Uppgift 2 (6p)

Antag att en processor använder 2-bitars dynamisk branch prediction. Från början står alla prediktorer i tillståndet *strongly taken*. Följande kod körs sedan på processorn:

```
add $s0, $zero, $zero      ; Initialize sum to 0
la $s1, arr                ; Get pointer to the beginning of arr
addi $s2, $s1, 40          ; Compute end of arr
loop_start:
beq $s1, $s2, loop_end     ; Test for loop termination
lw $t0, $s1                ; Load element from arr
addi $s1, $s1, 4           ; Increment array pointer for next iteration
slti $t1, $t0, 0           ; Check if loaded element is negative
beq $t1, $zero, loop_start ; If not, continue the loop
add $s0, $s0, $t0          ; Add element to sum of negative elements
j loop_start               ; Jump to beginning of loop
loop_end:
```

Koden löper igenom arrayen `arr` och summerar alla negativa element i den. `arr` innehåller följande 10 värden (uppräknade i den ordning de ligger i arrayen):

```
{ 3, 8, -5, 10, 7, -6, -5, 18, 20, 1 }
```

Hur många korrekt respektive felaktigt förutsagda hopp kommer att inträffa under körningen av programmet, om man antar att alla villkorliga hoppinstruktioner associeras till *olika* prediktorer?

1 Del III - Minneshierarkier (10p)

Uppgift 1 (6p)

En cache är given som rymmer 4 kB (4096 bytes) data (metadata så som tags, valid-bitar osv. ej inräknade) och har en blockstorlek på 64 bytes. En serie minnesaccesser görs till följande 16-bitars adresser (se nästa blad):

1. 1001110100101110
2. 0110010100000110
3. 0110010100101110
4. 1101100100001101
5. 0001110100101000
6. 1001010100000110

Adresserna anges binärt och adresserar på byte-nivå. Säg att cachén är helt tom innan minnesaccesserna ovan görs. Hur kommer cachén att se ut efter den sista minnesaccessen om cachén är

- (a) direktmappad? (3 poäng)
- (b) 4-vägs set-associativ och använder LRU som utbytesstrategi? (3 poäng)

Ange index, valid-bit och tag för de platser i cachén som har uppdaterats (övriga platser behöver inte redovisas). Redogör även för vilka accesser som missar i cachén och hur cachén uppdateras vid dessa missar.

Uppgift 2 (4p)

Nedan ges parametrar för några olika cache-organisationer.

	Total storlek	Blockstorlek	Antal vägar(ways)	Antal set	Benämning
a	32 KB	64	?	128	?
b	?	32	1	256	?
c	?	32	64	?	fullt associativ
d	16 KB	32	?	256	?

Din uppgift är att fylla i de parametrar som saknas (markerade med frågetecken). Med "benämning" menas det begrepp man brukar använda för organisationen i fråga. Observera att den totala cache-storleken inte räknar in storleken på metadata, dvs. valid-bitar, LRU-bitar etc. Observera dessutom att $1 \text{ KB} = 2^{10} = 1024 \text{ bytes}$. Motivera dina svar.

Del IV - Mikroprogrammering (10p)

Uppgift 1: Absolutbelopp (7p)

Skriv exekveringsfasen av instruktionen ABS, "ABSolute value". Instruktionen ska stödja alla de adresseringsmoder som stöds av t.ex. instruktionen ADDC, och ska räkna ut absolutbeloppet av operanden och lagra det i ackumulatorregistret. Ge ditt svar som

- (a) Register-Transfer Notation.
- (b) mikrokod.

Du kan placera mikrokoden var du vill i mikrominnet. Som stöd kan du använda sid. 38 från mikro-programmeringskompendiet som bifogas denna tentamen.

Uppgift 2: Ellens adresseringsmoder(3p)

Antag att man initialt har följande minneslayout.

Adress	Innehåll
10	12
11	33
12	56
13	7C
14	AA

Dessutom har index-registret värdet 11. Alla värden anges hexadecimalt.

Vad laddar följande Ellen-instruktioner in för värden i ackumulatorregistret?

- a) LOAD #\$14
- b) LOAD \$11
- c) LOAD \$2(X)

ASSEMBLER-INSTRUKTIONER OCH MASKINKOD (HEX-FORMAT) I ELLEN.

Mnemonics	Implied	Immediate #	Absolute	Indexed (X)	Relative (PC)	Flags NZCV	Comment
LOAD	-	00	02	04	06	** - -	m(ea)->AR
STORE	-	-	0A	0C	0E	** - -	ar->M(ea)
ADDC	-	10	12	14	16	****	ar+m(ea)+c->AR
SUBC	-	18	1A	1C	1E	****	ar-m(ea)-NOT(c)->AR
CLC	27	-	-	-	-	-- 0-	0->C
SEC	2F	-	-	-	-	-- 1-	1->C
JUMP	-	-	32	34	36	----	ea->PC (adr->PC)
INX	4F	-	-	-	-	****	x+1->X
DEX	57	-	-	-	-	** - -	x-1->X
LDX	-	58	5A	5C	5E	----	m(ea)->X
BNE	-	-	62	64	66	----	ea->PC om Z=0, Branch if Not zero
BEQ	-	-	6A	6C	6E	----	ea->PC om Z=1, Branch if Equal to zero
BCC	-	-	72	74	76	----	ea->PC om C=0, Branch if Carry Clear
BCS	-	-	7A	7C	7E	----	ea->PC om C=1, Branch if Carry Set
LSP	-	80	82	84	86	----	m(ea)->SP
PUSH	8F	-	-	-	-	----	ar->M(sp), sp-1->SP
PULL	97	-	-	-	-	----	sp+1->SP, m(sp)-> AR
JSR	-	-	9A	9C	9E	----	pc->M(sp), sp-1->SP, ea->PC
RTS	A7	-	-	-	-	----	sp+1->SP, m(sp)->PC
CMP	-	A8	AA	AC	AE	****	ar-m(ea)
CPX	-	B0	B2	B4	B6	****	x-m(ea)

Figur 25 Assemblerinstruktioner och maskinkod i ELLENBeteckningar iFigur 25:

Stora bokstäver: AR, PC, SP, X = registret (M=huvudminnet).

Små bokstäver: ar, pc, sp, m(ea) = innehållet i registret eller huvudminnesordet med adress ea
ea = effektiva adressen.Flags (Statusregister):

N= bit 7 i resultatet

Z=1 om resultatet = 0, annars är Z=0

V=1 vid spill, annars är V=0

C=1 om carry vid addition, annars blir C=0 eller

C=0 om borrow vid subtraktion, annars blir C=1

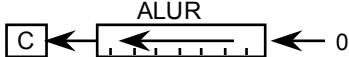


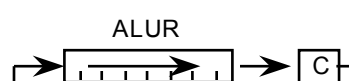
* Flaggan ställs enligt resultatet av assemblerinstruktionen.

- Flaggan påverkas inte av assemblerinstruktionen.

1.21 Mikroinstruktionsformat

Antal bitar	$\mu M1$			$\mu M2$			$\mu M3$
	TB	FB	BK	ALU	μPC	pc+1	μ hoppadress
	3	3	2	4	3	1	8

1.22 Mikroinstruktionskoder för olika fält - sammanfattning

TB = Sändarregister vid Bussöverföring	000 001 010 011	ADR = AdressRegistret ALUR = ALU Registret PC = Programräknaren (ProgramCounter) M = HuvudMinnet, OBS måste anropas 3 gånger i följd för att ge svar från minnet.
FB = Mottagarregister vid Bussöverföring	100 101 110 111	IR = InstruktionsRegistret. Kan användas både som TB och FB SP = StackPekare X = indeXregistret AR= AckumulatorRegistret
BK-Fältet Buss Kontroll	00 01 10 11	Bussen avstängd, data på bussen odefinierat. Ingen överföring till registret enligt FB-fältet. tb -> FB. Innehållet i register enligt TB kopieras till register enligt FB. tb + 1 -> FB. Som 01 men data ökas med 1 vid kopieringen. Innehållet i register enligt TB ändras ej tb - 1 -> FB. Som 10 men kopians värde minskas med 1.
ALU-fältet OBS!! Dessa funktioner fungerar bara om FB fältet innehåller koden 001	0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111	ALU transparent. Talet på databussen läses ner i ALUR. 0 -> ALUR. ALUR nollställs ADC ADd with Carry. alur + tb + c -> ALUR SUBC SUB with Carry. alur - tb - NOT(c) -> ALUR CC Clear Carry. 0 -> C. SC Set Carry. 1 -> C. AND logical AND. alur AND tb -> ALUR OR logical OR. alur OR tb -> ALUR XOR logical XOR. alur XOR tb -> ALUR 1001 ASL Algorithmic Shift Left  1010 LSR Logic Shift Right  1011 RLC Rotate Left through Carry  1100 RRC Rotate Right through Carry  1101 INV Bilda 1-komplementet av alur, lagra i ALUR (invertera). 1110 Spara statusregistret internt i CPU (ej på stacken!). 1111 Återställ statusregistret från den interna lagringsplatsen.
μPC -fältet	000 001 010 011 100 101 110 111	$\mu pc + 1 \rightarrow \mu PC$ OP-fältet via K1 -> μPC AM-fältet via K2 -> μPC Villkorligt hopp ($\mu m3 \rightarrow \mu PC$) om C = 1, annars $\mu pc + 1 \rightarrow \mu PC$ Villkorligt hopp ($\mu m3 \rightarrow \mu PC$) om V = 1, annars $\mu pc + 1 \rightarrow \mu PC$ Villkorligt hopp ($\mu m3 \rightarrow \mu PC$) om N = 1, annars $\mu pc + 1 \rightarrow \mu PC$ Villkorligt hopp ($\mu m3 \rightarrow \mu PC$) om Z = 1, annars $\mu pc + 1 \rightarrow \mu PC$ Ovillkorligt hopp, dvs. $\mu m3 \rightarrow \mu PC$, alltid