Tentamen i kursen Datorsystemteknik (EDA330/EDA370/EDA440 för D/E/IT)

Datorsystemteknik för D/E/IT

24/5 2003

Tentamensdatum: Lördag 24/5 2003 kl. 14.15 i sal M

Examinatorer: Peter Folkesson och Jonas Vasell

Institution: Datorteknik

Förfrågningar: Peter Folkesson (ankn. 1676)

Lösningar: anslås måndag 26/5 på institutionens anslagstavla utanför laboratoriet och kursens hemsida på Internet

Resultat: anslås senast torsdag 12/6 på institutionens anslagstavla utanför laboratoriet och kursens hemsida på Internet

Rättningsgranskning: tid och plats anslås tillsammans med resultaten

Betygsgränser: 3: 24-35 poäng, 4: 36-47 poäng, 5: 48-60 poäng

Tillåtna hjälpmedel: Typgodkänd kalkylator

Allmänt: För uppgift 1-3 behöver endast svar anges. Felaktiga svar på deluppgifter till uppgift 1 och 2 ger *minuspoäng*, dock är minsta poäng på hela uppgiften alltid 0. För uppgift 3 kan även felaktiga svar ge pluspoäng, för korrekt poängbedömning av felaktiga svar till uppgift 3 fordras dock redovisade uträkningar.
För full poäng på övriga uppgifter krävs både ett korrekt svar och en *motivering*. En bra motivering är minst lika viktig som ett korrekt svar. Redovisa noggrannt alla gjorda antaganden utöver de som anges i uppgiftstexten.
Skriv tydligt och använd gärna figurer. Maximal poäng på varje uppgift anges inom parentes efter uppgiftstexten.

Lycka till!

Svarstalong uppgift 1-3:

Deluppgift	1	X	2	Poäng
1 a)				
1 b)				
1 c)				
1 d)				
1 e)				
1 f)				
1 g)				
1 h)				
1 i)				
1 j)				
1 k)				
1 l)				
2 a)				
2 b)				
2 c)				
2 d)				
3 a)				
3 b)				

3 c)	Steg:																			
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Namn, personnummer: _____

Uppgifter (1-4):

1. Nedan följer ett antal frågor med tre svarsalternativ (1, X, 2) vardera, varav *endast ett* är rätt. Ställ upp svaren som en tipsrad. Använd svarstalongen på sidan 2. Varje rätt svar ger ett pluspoäng och **varje felaktigt svar ger ett minuspoäng**. Inget svar ger noll poäng. Minsta poäng på hela uppgiften är noll. (12 p)

   a. MIPS-instruktionen `lui` (1) laddar den minst signifikanta delen av en stor konstant. (X) laddar ett teckenlöst heltal. (2) laddar den mest signifikanta delen av en stor konstant.
   b. En modern ISA karakteriseras av (1) många minnesoperander. (X) få instruktionsformat. (2) få generella register.
   c. Antalet instruktioner som krävs för att utföra ett program på en viss processor beror på (1) vilken ISA som används. (X) hur minnet är organiserat. (2) vilken hårdvaruteknologi som används.
   d. I en pipelinad processor (1) är genomströmningshastigheten låg. (X) krävs likformig exekveringscykel. (2) tar en instruktion alltid en klockcykel att genomföra.
   e. Vilken av följande hopp-predikteringsstrategier brukar minska effekterna av styrkonflikter i pipelinen bäst? (1) *Assume not taken*. (X) *Assume taken*. (2) *History based*.
   f. Med hög rumslokalitet i minnessystem avses att (1) om en adress refereras så är det stor sannolikhet att en närliggande adress snart refereras. (X) om en adress refereras så är det stor sannolikhet att den snart refereras igen. (2) få adresser refereras.
   g. TLB (1) är en komponent som buffrar operationer i flyttalsenheter. (X) är en komponent som lagrar en del av sidtabellen för virtuellt minne. (2) är en komponent för grafikacceleration.
   h. För cache-minnen betyder fullt associativ att (1) varje block kan lagras på exakt ett ställe i cache-minnet. (X) varje block kan lagras var som helst i cache-minnet. (2) varje cache-block är associerat med ett primärminnesblock.
   i. Handskakningsprotokoll krävs i allmänhet för (1) synkrona bussar. (X) asynkrona bussar. (2) både asynkrona och synkrona bussar.
   j. Vilket av följande är absolut nödvändigt om man vill använda sig av bussar med flera *bus masters*. (1) Handskakning. (X) Arbitrering. (2) Multiplexing.
   k. Det största talet som kan representeras med dubbel precision (IEEE 754 double precision) är c:a $2.0*10^{308}$. Vilket är det största tal som kan representeras med enkel precision (IEEE 754 single precision)? (1) $2.0*10^{38}$. (X) $2.0*10^{154}$. (2) $1.0*10^{308}$.
   l. Skillnaden i snabbhet mellan minnen och processorer tenderar (1) att minska. (X) att vara oförändrad. (2) att öka.

2. Nedan följer ett antal frågor med tre svarsalternativ (1, X, 2) vardera, varav *endast ett* är rätt. Ställ upp svaren som en tipsrad. Använd svarstalongen på sidan 2. Varje rätt svar ger 3 pluspoäng och **varje felaktigt svar ger 1 minuspoäng**. Inget svar ger noll poäng. Minsta poäng på hela uppgiften är noll. (12 p)

   a. Betrakta följande MIPS-program:

```
        lw $1, 800($5)
        sub $2, $2, $1
        slti $4, $2, 100
        beq $4, $0, Exit
        :
Exit:   add $9, $9, $6
        :
```

   Antag att programmet exekveras med den pipeline som visas i bilaga 1 och att hoppet utförs. Antag vidare att ingen hantering av pipelinekonflikter byggts in. Ange hur många `nop`-instruktioner (som inte gör något) som måste läggas in i programmet för att det ska fungera som avsett. Registerfilen tillåter ej att data läses och skrives till samma register samtidigt.
   (1) 6 stycken (X) 10 stycken (2) 12 stycken

   b. Hur många `nop`-instruktioner kan man ta bort i föregående deluppgift om data forwarding (bypassing) införs?
   (1) 3 stycken (X) 5 stycken (2) 8 stycken

   c. Ett visst program kräver 3 s CPU-tid att exekvera på MIPS med ett idealt minnessystem där alla minnesreferenser endast tar en klockcykel (dvs inga pipeline stalls). CPI är 1,8, och processorns klockfrekvens är 600 MHz. Antag att minnessystemet ändras, så att ett cacheminne med 90% träffsannolikhet (hit rate) och 12 klockcyklers straffkostnad för missar (miss penalty) införs för alla dataåtkomster (instruktionshämtningen går fortfarande på en klockcykel). CPU-tiden för programmet ökar då med ca. 1 s. Hur många dataåtkomster gjordes i minnessystemet under programexekveringen?
   (1) ca. 500 miljoner (X) ca. 750 miljoner (2) ca. 1 miljard

   d. Hur bred är en tag i ett 256 KB stort cacheminne med associativitet 4 och 64 byte stora block där uppslagningar sker med 32 bitars byte-adresser?
   (1) 12 bitar (X) 16 bitar (2) 19 bitar

3. Nedan följer ett antal frågor där endast svar behöver anges. Varje rätt svar ger full poäng men även felaktiga svar kan ge poäng (ej minuspoäng). Använd svarstalongen på sidan 2. För korrekt poängbedömning av felaktiga svar kan redovisning på separata papper fordras.

   a. Systembussen i en styrdator är 32 bitar bred med multiplexad överföring av adresser och data, har centraliserad parallell arbitrering (tar minst 2 busscykler att reservera), och är synkron med 100 MHz klockfrekvens. Varje överföring av 32 bitar över bussen tar en klockcykel. Till systembussen är systemets primärminne och processor samt ett nätverkskort kopplade. Processorn är av den typ som visas i bilaga 1 med 200 MHz klockfrekvens, och innehåller separata instruktions- och datacache med blockstorleken 4 ord som använder

systembussen vid cachemissar. Vid en miss i endera cacheminnet stoppas processorn tills hela det saknade blocket hämtats in från primärminnet. En träff i cacheminnet avslutas på en cykel.

Primärminnet är av DRAM-typ och producerar block om upp till 4 par av 32-bitars ord för varje given blockadress. Antalet ordpar kan ställas in för varje åtkomst. Efter att en blockadress givits till minnet tar det 50 ns tills de första två orden är tillgängliga, och sedan tar det ytterligare 15 ns för varje ytterligare därpå följande par av ord att läsas ut. Minnesåtkomst av efterföljande ord får överlappa med överföring av tidigare ord. Före varje blockläsning reserveras bussen för hela läsningen, och släpps fri för andra överföringar först när hela blockläsningen är klar.

Missannolikheten för instruktionscacheminnet har vid mätningar visat sig vara 2%. CPI utan inverkan av instruktionsmissar har uppmätts till 1,6. Du kan anta att vid arbitrering av bussen ges högst prioritet åt instruktionscacheminnet. Arbitreringen sker parallellt med blocköverföringarna så att direkt en överföring är klar så kan nästa påbörjas. Vad är belastning på systembussen (andel av tiden som bussen är reserverad) pga instruktionsmissar? (6 p)

b. Följande bitmönster representerar två instruktioner som sammantaget bildar en pseudoinstruktion (en instruktion som egentligen inte finns i instruktionsuppsättningen men som av assemblatorn kan översättas till befintliga instruktioner). Vilken pseudoinstruktion avses? (4 p)

$$00000001010010010000100000101010_2$$
$$00010000001000000000010000000000_2$$

c. Beskriv vad som händer vid ett sidfel som uppstår vid en viss minnesläsning genom att välja ut och ordna händelser från följande lista. Ta med alla händelser fram tills det att minnesläsningen avslutats. *Alla händelser ska inte vara med, och vissa händelser kan vara med flera gånger.* Observera också att det finns saker som händer eller kan hända, men som inte tas med i listan. Dessa ska ignoreras. Fyll i svaret på den bifogade svarsblanketten genom att för varje steg i ordningen ange bokstaven för den händelse som inträffar. (6 p)

A. Uppslagning i cache ger en träff.
B. Uppslagning i cache ger en miss.
C. Uppslagning sker i TLB, som signalerar en miss.
D. Uppslagning sker i TLB, som ger en korrekt översättning till fysisk adress.
E. Avbrottshanteraren lagrar undan CPU- och processtillstånd.
F. Kontroll sker att den önskade sidåtkomsten är tillåten.
G. En plats i primärminnet för den sökta sidan väljs ut med hjälp av en utbytesalgoritm.
H. Programmet där sidfelet uppstod fortsätter sin exekvering med uppdaterad TLB.
I. Minnesläsningen påbörjas.
J. Avbrottshanteraren gör återhopp till minnesläsningsinstruktionen.
K. Minnesinnehållet returneras från cache till CPU.
L. Överföring av sidan till primärminnet med hjälp av DMA inleds.

M. Sidtabellen uppdateras.
N. CPU avbryts av en signal att DMA av sidan är klar.
O. Uppslagning sker genom hårdvara i MMU i sidtabellen i primärminnet.
P. Den virtuella adressen delas upp i virtuellt sidnummer och offset.
R. Det visar sig att sidan inte finns i primärminnet, och en avbrottssignal skickas till CPU.
S. Avbrottshanteraren upptäcker att sidfel uppstått och anropar en rutin för hantering av sidfel.
T. CPU börjar exekvera en avbrottshanterare.

4. För full poäng på följande uppgifter krävs både ett korrekt svar och en *motivering*. En bra motivering är minst lika viktig som ett korrekt svar. Redovisa noggrannt alla gjorda antaganden utöver de som anges i uppgiftstexten.

   a. Antag att sannolikheten för sidfel vid dataåtkomster är cirka 0,00001% för ett program som kör på en typisk persondator från 2002/2003. Är det rimligt att inverkan av cachemissar vid dataåtkomster på programmets exekveringstid kan bli av samma storleksordning som inverkan av sidfel? Gör en överslagsberäkning med rimliga antaganden om de parametrar som behövs för uträkningen. (3 p)

   b. Vad kan orsaka att ett programs exekveringstid varierar även om det kör i ett och samma datorsystem varje gång? Ge minst tre olika skäl. (3 p)

5. Antag att en MIPS processor ska kopplas till primärminnet (DRAM) via en processor-minnesbuss. Denna buss är synkron med frekvensen 100 MHz och tillåter multiplexad överföring av ett ord (32 bitar) data eller adress varje busscykel. För varje adress kan antingen 1 eller 8 ord överföras. Primärminnets åtkomsttid får anses försumbar.

   Följande operationstider är kända för processorns ingående komponenter:
   ALU (klarar heltals add/sub/mul/div samt logiska operationer): 2 ns
   Registerfil läsning/skrivning: 1 ns för både heltal och flyttal (registerfilen tillåter läsning samtidigt som skrivning)
   Flyttalsmultiplikation/flyttalsdivision: 12 ns
   Övriga flyttalsoperationer: 6 ns
   Operationstiden för övriga komponenter i processorn får försummas.
   Som benchmark används ett program med följande instruktions-mix:

flyttalsmultiplikationer exekveras:	10000 ggr
flyttalsadditioner exekveras:	6000 ggr
flyttalsdivisioner exekveras:	2000 ggr
villkorliga hopp-instruktioner (branch) exekveras:	10000 ggr
övriga operationer exekveras:	72000 ggr

   10% av alla instruktioner skriver till minnet (sw), 20% läser data från minnet (lw). Programmet innehåller inga ovillkorliga hopp (jump).

   a. Vad blir exekveringstiden för programmet om man väljer en encykelimplementering av CPU’n utan cache? (4 p)

b. Vad blir exekveringstiden för programmet om man istället väljer en flercykelimplementering av CPU'n utan cache med cykeltiden 2 ns? (4 p)

c. Vad blir exekveringstiden för programmet om man istället väljer en pipelineimplementering av CPU'n liknande den i bilaga 1 med separat instruktions- och datacache, båda med en operationstid för läsning/skrivning på 1 ns? Instruktionscachen har en träffsannolikhet på 99%. Datacachen är av write-back typ och har träffsannolikheten 95%. Varje transaktion på processor-minnesbussen innebär nu att data motsvarande ett block som omfattar 8 ord, dvs. ett block i cachen, läses eller skrivs till primärminnet. Endast flyttalsoperationer och cachemissar ger upphov till pipeline-stalls medan övriga pipelinekonflikter hanteras av hårdvaran. Cykeltiden är fortfarande 2 ns. (6 p)

TREVLIG SOMMAR!

Bilaga 1: MIPS maskininstruktioner och pipeline

Common MIPS instructions.

Notes: *op*, *funct*, *rd*, *rs*, *rt*, *imm*, *address*, *shamt* refer to fields in the instruction format. The program counter PC is assumed to point to the next instruction (usually 4 + the address of the current instruction). M is the byte-addressed main memory.

Assembly instruction	Instr. format	*op* *op/funct*	Meaning	Comments
`add $rd, $rs, $rt`	R	0/32	`$rd = $rs + $rt`	Add contents of two registers
`sub $rd, $rs, $rt`	R	0/34	`$rd = $rs - $rt`	Subtract contents of two registers
`addi $rt, $rs, imm`	I	8	`$rt = $rs + imm`	Add signed constant
`addu $rd, $rs, $rt`	R	0/33	`$rd = $rs + $rt`	Unsigned, no overflow
`subu $rd, $rs, $rt`	R	0/35	`$rd = $rs - $rt`	Unsigned, no overflow
`addiu $rt, $rs, imm`	I	9	`$rt = $rs + imm`	Unsigned, no overflow
`mfc0 $rt, $rd`	R	16	`$rt = $rd`	*rd* = coprocessor register (e.g. epc, cause, status)
`mult $rs, $rt`	R	0/24	`Hi, Lo = $rs * $rt`	64 bit signed product in Hi and Lo
`multu $rs, $rt`	R	0/25	`Hi, Lo = $rs * $rt`	64 bit unsigned product in Hi and Lo
`div $rs, $rt`	R	0/26	`Lo = $rs / $rt, Hi = $rs mod $rt`	
`divu $rs, $rt`	R	0/27	`Lo = $rs / $rt, Hi = $rs mod $rt` (unsigned)	
`mfhi $rd`	R	0/16	`$rd = Hi`	Get value of Hi
`mflo $rd`	R	0/18	`$rd = Lo`	Get value of Lo
`and $rd, $rs, $rt`	R	0/36	`$rd = $rs & $rt`	Logical AND
`or $rd, $rs, $rt`	R	0/37	`$rd = $rs | $rt`	Logical OR
`andi $rt, $rs, imm`	I	12	`$rt = $rs & imm`	Logical AND, unsigned constant
`ori $rt, $rs, imm`	I	13	`$rt = $rs | imm`	Logical OR, unsigned constant
`sll $rd, $rs, shamt`	R	0/0	`$rd = $rs << shamt`	Shift left logical (shift in zeros)
`srl $rd, $rs, shamt`	R	0/2	`$rd = $rs >> shamt`	Shift right logical (shift in zeros)
`lw $rt, imm($rs)`	I	35	`$rt = M[$rs + imm]`	Load word from memory
`sw $rt, imm($rs)`	I	43	`M[$rs + imm] = $rt`	Store word in memory
`lbu $rt, imm($rs)`	I	37	`$rt = M[$rs + imm]`	Load a single byte, set bits 8-31 of $*rt* to zero
`sb $rt, imm($rs)`	I	41	`M[$rs + imm] = $rt`	Store byte (bits 0-7 of $*rt*) in memory
`lui $rt, imm`	I	15	`$rt = imm * 2^16`	Load constant in bits 16-31 of register $*rt*
`beq $rs, $rt, imm`	I	4	`if($rs==$rt) PC = PC + imm` (PC always points to next instruction)	
`bne $rs, $rt, imm`	I	5	`if($rs!=$rt) PC = PC + imm` (PC always points to next instruction)	
`slt $rd, $rs, $rt`	R	0/42	`if($rs<$rt) $rd = 1; else $rd = 0`	
`slti $rt, $rs, imm`	I	10	`if($rs<imm) $rt = 1; else $rt = 0`	
`sltu $rd, $rs, $rt`	R	0/43	`if($rs<$rt) $rd = 1; else $rd = 0` (unsigned numbers)	
`sltiu $rt, $rs, imm`	I	11	`if($rs<imm) $rt = 1; else $rt = 0` (unsigned numbers)	
`j destination`	J	2	`PC = address*4`	Jump to *destination*, *address* = *destination*/4
`jal destination`	J	3	`$ra = PC; PC = address*4` (Jump and link, *address* = *destination*/4)	
`jr $rs`	R	0/8	`PC = $rs`	Jump to address stored in register $*rs*

MIPS registers

Name	Number	Usage
`$zero`	0	constant 0
`$at`	1	reserved for assembler
`$v0 - $v1`	2-3	expression evaluation and function results
`$a0 - $a3`	4-7	arguments
`$t0 - $t7`	8-15	temporary, saved by caller
`$s0 - $s7`	16-23	temporary, saved by called function
`$t8 - $t9`	24-25	temporary, saved by caller
`$k0 - $k1`	26-27	reserved for kernel (OS)
`$gp`	28	points to middle of a 64K block in the data segment
`$sp`	29	stack pointer (top of stack)
`$fp`	30	frame pointer (beginning of current frame)
`$ra`	31	return address
`Hi, Lo`	-	store partial result of mult and div operations
`PC`	-	contains the address of the next instruction to be fetched (not real register, only to define instructions)
`Status`	-	register 12 in coprocessor 0, stores interrupt mask and enable bits
`Cause`	-	register 13 in coprocessor 0, stores exception type and pending interrupt bits
`Epc`	-	register 14 in coprocessor 0, stores address of instruction causing exception

MIPS Instruction formats

Format	Bits 31-26	Bits 25-21	Bits 20-16	Bits 15-11	Bits 10-6	Bits 5-0
R	op	rs	rt	rd	shamt	funct
I	op	rs	rt	imm		
J	op	address				

MIPS Assembler syntax

```
                                # This is a comment
        .data                   # Store following data in the data
                                # segment
items:                          # This is a label connected to the
                                # next address in the current segment
        .word 1, 2              # Stores values 1 and 2 in next two
                                # words
hello:  .asciiz "Hello"         # Stores null-terminated string in
                                # memory
        .text                   # Store following instructions in
                                # the text segment
main:   lw $t0, items($zero)    # Instruction that uses a label to
                                # address data
```

MIPS Pipeline