Tentamen i kursen Datorsystemteknik (EDA330 för D och EDA370 för E)

# Datorsystemteknik för D/E

16/1 2003

**Tentamensdatum**: Torsdag 16/1 2003 kl. 8.45 i sal M

**Examinatorer**: Peter Folkesson och Jonas Vasell

**Institution**: Datorteknik

**Förfrågningar**: Peter Folkesson (ankn. 1676)

**Lösningar**: anslås fredag 17/1 på institutionens anslagstavla utanför laboratoriet och kursens hemsida på Internet

**Resultat**: anslås senast fredag 31/1 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 uppgift 4 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) | | | | |
| 2 e) | | | | |
| 2 f) | | | | |
| 2 g) | | | | |
| 2 h) | | | | |
| 2 i) | | | | |
| 3 a) | | | | |
| 3 b) | | | | |
| 3 c) | | | | |
| 3 d) | | | | |
| 3 e) | | | | |
| 3 f) | | | | |

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-arkitekturen tillåter adressering av (1) $2^{32}$ ord. (X) $2^{30}$ ord. (2) $2^{30}$ byte.
   b. En superskalär processor (1) kan starta exekvering av flera instruktioner samtidigt. (X) har en extra lång pipeline. (2) har en speciellt kraftfull ALU.
   c. Det vanligaste sättet att hantera resurskonflikter i en pipeline är att (1) stanna delar av pipelinen tills konflikten löses upp. (X) tidigarelägga tillgång till resultat. (2) förutsäga resultatet av exekveringen.
   d. TLB är en komponent (1) som buffrar operationer i flyttalsenheter. (X) som lagrar en del av sidtabellen för virtuellt minne. (2) för grafikacceleration.
   e. För cache-minnen betyder direktavbildat (*direct mapped*) att (1) varje cache-block är associerat med ett primärminnesblock. (X) varje block kan lagras var som helst i cache-minnet. (2) varje block kan lagras på exakt ett ställe i cache-minnet.
   f. DMA står för (1) Dual Memory Access. (X) Direct Memory Access. (2) Dynamic Memory Access.
   g. Den typiska överföringshastigheten (*transfer rate*) för hårddiskar ligger idag omkring (1) 150 KB/s. (X) 15 MB/s. (2) 150 MB/s.
   h. En processors klockfrekvens beror av (1) hårdvaruteknologi och processororganisation. (X) instruktionsuppsättning och processororganisation. (2) hårdvaruteknologi och program.
   i. Kapaciteten för de största DRAM-kretsarna ökar på tre år ungefär med en faktor (1) 2. (X) 3. (2) 4.
   j. 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*)? C:a (1) $2.0*10^{38}$. (X) $2.0*10^{154}$. (1) $1.0*10^{308}$.
   k. De minsta detaljer som kan implementeras med VLSI beräknas för 2010 vara c:a (1) 0,07 µm. (X) 0,007 µm. (2) 0,0007 µm.
   l. *Write-invalidate* och *write-update* är exempel på (1) utbytesstrategier för virtuella sidor.(X) tekniker för att lösa pipelinekonflikter. (2) metoder för att uppnå cache-koherens i parallelldatorsystem.

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 2 pluspoäng och **varje felaktigt svar ger 2 minuspoäng**. Inget svar ger noll poäng. Minsta poäng på hela uppgiften är noll. (18 p)

   a. Beräkna hur många cykler följande MIPS-program tar att exekvera med den pipeline som visas i bilaga 1 om *assume not taken* används för att hantera styrkonflikterna:

```
      addi $5, $1, 16
L1:   lw   $4, 0($5)
      addi $5, $5, -4
      add  $3, $3, $4
      add  $3, $3, $3
      bne  $5, $1, L1
      sw   $3, 0($5)
```

Utgå från att eventuella datakonflikter hanteras med hjälp av forwarding, och att de därför inte ger upphov till några stalls. Räkna cykeln när första instruktionen hämtas som cykel ett, och svara med numret på den cykel då den sista instruktionen utförs i sista pipeline-steget.
(1) 29 cykler (X) 35 cykler (2) 38 cykler

b. Hur bred är en tag i ett 32 KB stort cacheminne med associativitet 4 och 32 B stora block där uppslagningar sker med 32 bitars byte-adresser?
(1) 16 bitar (X) 18 bitar (2) 19 bitar

c. Antag ett icke-associativt (direct mapped) cacheminne med totala storleken 1 KB. För detta gäller under mätningar vid exekvering av ett visst program att miss-frekvensen är 38% om blockstorleken sätts till 4 B, 22% om blockstorleken sätts till 16 B, 19% om blockstorleken sätts till 64 B. Miss penalty är 4+b/4 cykler, där b är blockstorleken, och det görs 1,2 minnesreferenser per instruktion. CPI utan cachemissar är 2, och klockfrekvensen påverkas inte av blockstorleken. Vilken av de tre blockstorlekarna bör man välja för att minimera exekveringstiden?
(1) 4 B (X) 16 B (2) 64 B

d. För ett skivminne gäller att det består av 4 dubbelsidiga skivor med ett läs/skriv-huvud per yta. Antalet cylindrar är 30000 och det går 500 sektorer per spår (*track*). Varje sektor rymmer 512 bytes och skivorna roterar med 7200 varv per minut. Söktiden (*seek time*) är 10 ms. Vad är skivminnets totala lagringskapacitet?
(1) 7,2 GB (X) 28,5 GB (2) 57 GB

e. Ett visst program kräver 45 ms CPU-tid att exekvera på MIPS med ett idealt minnessystem där alla minnesreferenser endast tar en klockcykel (dvs inga pipeline stalls). CPI är 1,5, och processorns klockfrekvens är 50 MHz. Hur många instruktioner exekveras i detta program?
(1) $1{,}5*10^6$ (X) $7{,}4*10^8$ (2) $1{,}7*10^9$

f. Om det binära ordet $10101111110000000100000000000000_2$ tolkas som ett heltal i tvåkomplementrepresentation, vilket heltal representerar ordet?
(1) $-821\times2^{14}$ (X) $-8217\times2^{14}$ (2) $-82175\times2^{14}$

g. Hur kan det binära ordet $10101100010000000000000000000000_2$ tolkas?
(1) Som flyttalet $3\times2^{-40}$. (X) Som MIPS-instruktionen `sw $0, 0($2)`.
(2) Som heltalet $335\times2^{22}$.

h. Antag att vi har ett datorsystem med följande karakteristik: Processorn adresserar virtuellt minne (kombinerat data- och instruktionsminne) med 32-bitars virtuella adresser. Det finns maximalt 1 GB fysiskt primärminne, och ett fyrvägs associativt cacheminne med kapacitet att lagra 32 KB data. Cacheminnets åtkomsttid vid träff är 5 ns. För sidöversättningar finns en fullt associativ TLB för 128 översättningar. TLBns åtkomsttid vid träff är 3 ns. Sidstorleken är 4 KB, och cacheminnets blockstorlek är 32 bytes. För såväl cacheminnet, TLBn som det virtuella minnet tillämpas write-back (copy-back) som skrivningsstrategi. Som utbytesalgoritm för cacheminnet och det virtuella minnet används LRU med 2-bitars tidsstämpel för varje block (sida). Virtuella sidor som tillhör operativsystemet är markerade med en flaggbit (protection bit) för att skyddas mot åtkomst från användarprocesser. Sekundärminnesadresser för sidor som inte finns i primärminnet lagras inte i sidtabellerna. Vad är det totala antalet bitar som cacheminnet måste kunna lagra?
(1) 279552 bitar (X) 281600 bitar (2) 283648 bitar

i. Vilken klockfrekvens kan processorn ha som bäst i uppgift 2 h) om man antar att en minnesåtkomst i normalfallet ska klaras av inom en processorklockcykel?
(1) 125 MHz (X) 200 MHz (2) 333 MHz

3. Nedan följer ett antal frågor där endast svar behöver anges. Varje rätt svar ger 3 poäng men även felaktiga svar kan ge poäng (ej minuspoäng). Använd svarstalongen på sidan 2. För korrekt poängbedömning av felaktiga svar fordras dock redovisade uträkningar på separata papper. (18 p)

a. Följande MIPS-program utför en beräkning som förekommer på många ställen i en viss tillämpning:

```
        slt   $t0,   $a0,   $a1
        beq   $t0,   $zero, L1
        sub   $s0,   $a1,   $a0
        beq   $zero, $zero, L2
L1:     sub   $s0,   $a0,   $a1
L2:     add   $v0,   $s0,   $zero
```

Om programmet utförs i en pipeline som den i bilaga 1, med stall som enda möjlighet att hantera pipelinekonflikter, hur många *extra* cykler orsakade av pipeline-konflikter kommer det att krävas för att exekvera programmet? Antag att $a0 från början innehåller värdet 3, och $a1 värdet 2 och att ett nytt registervärde kan läsas samtidigt som det skrivs.

b. I ett visst datorsystem är en CPU med klockfrekvensen 400 MHz och ett gränssnitt mot en systembuss kopplade till primärminnet (DRAM) via en processor-minnebuss. Denna buss är synkron med frekvensen 200 MHz och multiplexad överföring av ett ord (32 bitar) data eller adress varje busscykel. En överföring kan räknas som mottagen först i slutet av busscykeln.
En enhet (CPU eller systembussgränssnitt) som vill använda bussen för en transaktion skickar en begäran till en busstyrenhet som gör arbitrering och tidigast i påföljande busscykel skickar tillbaka ett tillstånd att använda bussen.

Om bussen är upptagen skickas tillståndet så snart bussen kommer att vara fri i påföljande busscykel. När enheten fått tillstånd att använda bussen kan den börja sin transaktion i busscykeln efter att tillståndet givits. Bussen hålls kvar av enheten till hela transaktionen är klar, men måste sedan släppas.
Varje transaktion på bussen innebär att data motsvarande ett block data som omfattar 16 byte läses från eller skrivs till primärminnet, vilket motsvarar ett block i CPUns inbyggda cache. Primärminnet behöver 50 ns för åtkomst av ett block efter att det fått en blockadress, och därefter kan orden i blocket överföras ett i taget varje busscykel. Förfarandet är oberoende av om det är en läsning eller skrivning, skillnaden är bara i vilken riktning data skickas. Vad är den maximala effektiva bandbredd som kan uppnås med processor-minnebussen?

c. På en arbetsstation byggd kring en MIPS-processor med 100 MHz klockfrekvens utförs en vetenskaplig beräkning uppdelad i en fast del och en iterativ del. Den fasta delen av beräkningen utförs bara en gång per körning och omfattar totalt $200\times10^6$ instruktioner varav 10% kräver dataminnesåtkomst. Den iterativa delen av beräkningen utförs 20 gånger. Varje iteration innebär att $2\times10^6$ instruktioner utförs, varav 40% kräver dataminnesåtkomst. CPI exklusive inverkan av minnesåtkomster är 1,5 för både den fasta och den iterativa delen. Processorn har två cacheminnen, ett för instruktioner och ett för data. För båda dessa gäller att kostnaden för missar (miss penalty) är 10 klockcykler. Vid en körning av beräkningen med UNIX som operativsystem ger tidmätning med ‘time’-kommandot (som ger användarprogrammets CPU-tid, operativsystemets CPU-tid, total tid, respektive andel CPU-tid av total tid) följande resultat:

```
4.2u 1.1s 0:07 75%
```

Inverkan av avbrott på beräkningens CPU-tid är försumbar. Vad är CPI för beräkningen (användarprogrammet) inklusive inverkan av minnesåtkomster?

d. Vad är missannolikheten för datacachen i uppgift 3 c) om missannolikheten för instruktionscachen är 1%?

e. Hur stor andel av den totala accesstiden måste i medeltal ägnas åt att söka upp rätt block om 8 KB stora block accessas på slumpmässiga ställen på skivminnet i uppgift 2 d)?

f. Vad är det totala antalet bitar som TLB i uppgift 2 h) måste kunna lagra?

4. Nedan beskrivs ett datorsystem med en MIPS-processor, cacheminne, MMU (inklusive TLB för virtuellt minne), och primärminne. Beräkna CPI för processorn utgående från de uppgifter som ges om systemet. (12 p)

Processorn är en MIPS implementerad som en 5-stegs pipeline med stegen Instruction Fetch, Instruction Decode, Execute, Memory Access, och Write Back. Forwarding används för att lösa upp datakonflikter så långt som möjligt. För alla typer av hopp används strategin Assume Not Taken, och hoppberäkningar (adress

och eventuellt villkor) görs i ID-steget. Processorns klockfrekvens är 100 MHz, och alla instruktioner spenderar normalt en klockcykel i varje pipeline-steg.

För de program som körs är följande statistik känd: Av alla exekverade instruktioner innebär 20% minnesläsningar, 10% minnesskrivningar, och 20% någon form av hopp. 70% av hoppen tas. 50% av minnesläsningarna följs direkt av en instruktion som använder det som läses in. Avbrott under programkörningen är så ovanliga att deras eventuella inverkan på CPI kan försummas. Effekter av uppstartning av pipelinen är också försumbara eftersom ett mycket stort antal instruktioner exekveras.

Systemet använder virtuellt minne, och alla adressöversättningar sker via en Memory Management Unit (MMU). För denna uppgift antas att alla adressöversättningar kan ske direkt i MMU utan sidfel eller missar i TLB. MMU behöver alltså inte kommunicera med primärminnet. De fysiska adresserna är 28 bitar breda.

Det finns två separata cacheminnen; ett för instruktioner (ICACHE), och ett för data (DCACHE). Båda arbetar med cacheblock som är 4 ord (=16 byte) stora. För ICACHE är träffsannolikheten 99%, och för DCACHE 90%. DCACHE använder write-back som skrivningsstrategi, och vid 50% av missarna i DCACHE måste ett cacheblock skrivas tillbaka till primärminnet. Cacheminnena använder fysiska adresser, och har en åtkomsttid vid träff på en processorcykel (inklusive adressöversättning). Vid en miss signalerar cacheminnet att det är klart så snart hela det saknade cacheblocket hämtats in.

MMU, ICACHE, och DCACHE (samt ett gränssnitt mot en systembuss) är kopplade till primärminnet via en processor-minnebuss. Denna buss är synkron, och har samma frekvens som processorn. Bussen är 32 bitar (=1 ord) bred, och har multiplexad adress- och dataöverföring. Ett ord kan överföras per busscykel. Varje transaktion på bussen innebär att data motsvarande ett cache-block läses från eller skrivs till primärminnet. En transaktion inleds med i genomsnitt en cykels väntan på att bussen ska bli ledig följd av en cykel för arbitrering. Så snart arbitreringen är klar tilldelas kontroll av bussen en av enheterna på bussen tills transaktionen är klar. Dataöverföringen inleds med att adressen till det aktuella blocket skickas till primärminnet. Primärminnet behöver 200 ns för åtkomst av första ordet i blocket, och 20 ns för de direkt efterföljande orden. Överföring på bussen och åtkomst i primärminnet kan ske parallellt. Förfarandet är oberoende av om det är en läsning eller skrivning, skillnaden är bara i vilken riktning data skickas.

TIPS: Identifiera först de olika typer av pipelinekonflikter som kan uppstå, och beräkna sedan inverkan av var och en av konflikttyperna på CPI. Du får poäng för varje korrekt identifierad konflikttyp, och för varje korrekt beräknad inverkan på CPI.

# SLUT

# 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 (tnot a real MIPS register, used 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