Tentamen i kursen Datorsystemteknik (EDA440 för IT)

Datorsystemteknik för IT

12/3 2003

Tentamensdatum: Onsdag 12/3 2003 kl. 8.45 i sal M

Examinatorer: Peter Folkesson och Jonas Vasell

Institution: Datorteknik

Förfrågningar: Peter Folkesson (ankn. 1676)

Lösningar: anslås torsdag 13/3 på institutionens anslagstavla utanför laboratoriet och kursens hemsida på Internet

Resultat: anslås senast måndag 31/3 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-instruktionen `slt` (1) skiftar ett tal åt vänster. (X) sätter lt-flaggan. (2) gör en jämförelse om ett tal är mindre än ett annat.
   b. Med *Big Endian* avses vilken ordning (1) de olika instruktionerna i en superskalär pipeline måste exekvera. (X) delarna av ord med större ordlängd än minnets ordstorlek måste lagras. (2) delarna av ord med större ordlängd än ALUn måste bearbetas vid aritmetiska operationer.
   c. En superskalär processor (1) kan starta exekvering av flera instruktioner samtidigt. (X) har en extra lång pipeline. (2) har en speciellt kraftfull ALU.
   d. Ett cacheminne bör implementeras med (1) DRAM-kretsar. (X) SRAM-kretsar. (2) Skivminne.
   e. Valet av utbytesalgoritm spelar störst roll för en (1) write-back cache. (X) fullt associativ cache. (2) direktavbildad cache.
   f. SCSI är en standard för (1) processor-minnesbussar. (X) systembussar. (2) I/O-bussar.
   g. För vilken typ av bussar är följande utmärkande egenskaper; standardiserad, variabel bandbredd, lång? (1) Systembuss. (X) I/O-buss. (2) Processor-minnebuss.
   h. Typisk storlek för ett processorchip idag är (1) 30 mm2. (X) 300 mm2. (2) 3000 mm2.
   i. En processors klockfrekvens beror av (1) hårdvaruteknologi och processororganisation. (X) instruktionsuppsättning och processororganisation. (2) hårdvaruteknologi och program.
   j. Booths algoritm är en metod för (1) flyttalsmultiplikation. (X) flyttalsdivision. (2) heltalsmultiplikation.
   k. Minneskoherens innebär (1) att alla tillgängliga kopior av en del av minnet alltid är lika. (X) att det bara får finnas en kopia av varje del av minnet. (2) att minnet är skrivskyddat.
   l. RISC (1) är anpassat för kompilatorer och VLSI teknik. (X) är anpassat till högnivåspråk och diskret implementering. (2) gynnar ”svåra fallet” och liten minnesåtgång.

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. Betrakta följande MIPS-program:

```
      lw $1, 100($2)
      sub $3, $3, $1
      slti $5, $3, 500
      beq $5, $0, L1
      :
L1:   add $6, $6, $8
      :
```

Antag att programmet exekveras med den pipeline som visas i bilaga 1, och att alla konflikter hanteras i hårdvaran genom pipeline stalling samt att ett nytt registervärde kan läsas först efter att det skrivits. Hur många klockcykler tar det från att den första instruktionen exekveras tills den sista lämnat WB-steget om hoppet tas?
(1) 16 cykler (X) 19 cykler (2) 21 cykler

b. Hur mycket kan antalet klockcykler minska jämfört med föregående deluppgift om data forwarding (bypassing) införs för att hantera datakonflikter där så är möjligt?
(1) 8 cykler (X) 10 cykler (2) 12 cykler

c. Hur bred är en tag i ett 64 KB stort cacheminne med associativitet 4 och 64 byte stora block där uppslagningar sker med 32 bitars byte-adresser?
(1) 16 bitar (X) 18 bitar (2) 20 bitar

d. 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 1GB fysiskt primärminne, och ett 4-vägs associativt cacheminne med kapacitet att lagra 64 KB data. För sidöversättningar finns en fullt associativ TLB för 128 översättningar. Sidstorleken är 16KB, och cacheminnets blockstorlek är 16 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. Hur stort minnesutrymme räknat i bytes behövs för att lagra en sidtabell om ett helt antal bytes används för att lagra varje rad i sidtabellen?
(1) 256 KB (X) 768 KB (2) 1 MB

e. Ett visst datorsystem är byggt kring en systembuss till vilken är kopplad en processor med en separat instruktions- och datacache, primärminne i form av DRAM, och ett eller flera DMA-gränssnitt till I/O-bussar. Systembussen är synkron med multiplexad överföring av ett ord (32 bitar) data eller adress varje busscykel med en frekvens av 166 MHz. Varje överföring till eller från minnet inleds med en cykel då adressen läggs ut på bussen. I de påföljande cyklerna sker sedan överföring av data ett ord i taget. För varje adress kan antingen ett eller 8 ord överföras. Vid DMA överförs alltid 8 ord åt gången på systembussen. Varje DMA-gränssnitt kan hantera två överföringar samtidigt,

men endast en DMA kan initieras åt gången och det tar 5 µs att initiera en ny DMA. I/O-bussarna är synkrona och överför ett ord i taget med en databandbredd på 50 MB/s. Till I/O-bussarna kopplas skivminnen med 5 ms genomsnittlig sök- och rotationstid, och en överföringsbandbredd på 10 MB/s. Processorn har klockfrekvensen 600 MHz och CPI=1,5. 10% av instruktionerna skriver ett ord till minnet, och 20% av instruktionerna läser ett ord från minnet. Instruktionscachen har en träffsannolikhet på 99.9%. Datacachen har en träffsannolikhet på 98% för läsningar, och för skrivningar tillämpas write-through av varje enskilt ord. Båda cacheminnena använder block om 8 ord. Vilken är den maximala databandbredden som systembussen kan klara?
(1) 332 MB/s (X) 590 MB/s (2) 664 MB/s

f. Hur stor andel av tiden är systembussen upptagen av trafik till och från cacheminnena i uppgift 2 e)?
(1) 21% (X) 37% (2) 59%

g. På en arbetsstation byggd kring en MIPS-processor med f=500 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 miljoner instruktioner varav 25% kräver dataminnesåtkomst. Den iterativa delen av beräkningen utförs N=100 gånger. Varje iteration innebär att 20 miljoner 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:

```
8.4u 2.2s 0:14 75%
```

Inverkan av avbrott på beräkningens CPU-tid är försumbar. Vad är CPI för beräkningen (användarprogrammet) inklusive inverkan av minnesåtkomster?
(1) 1,51 (X) 1,71 (2) 1,91

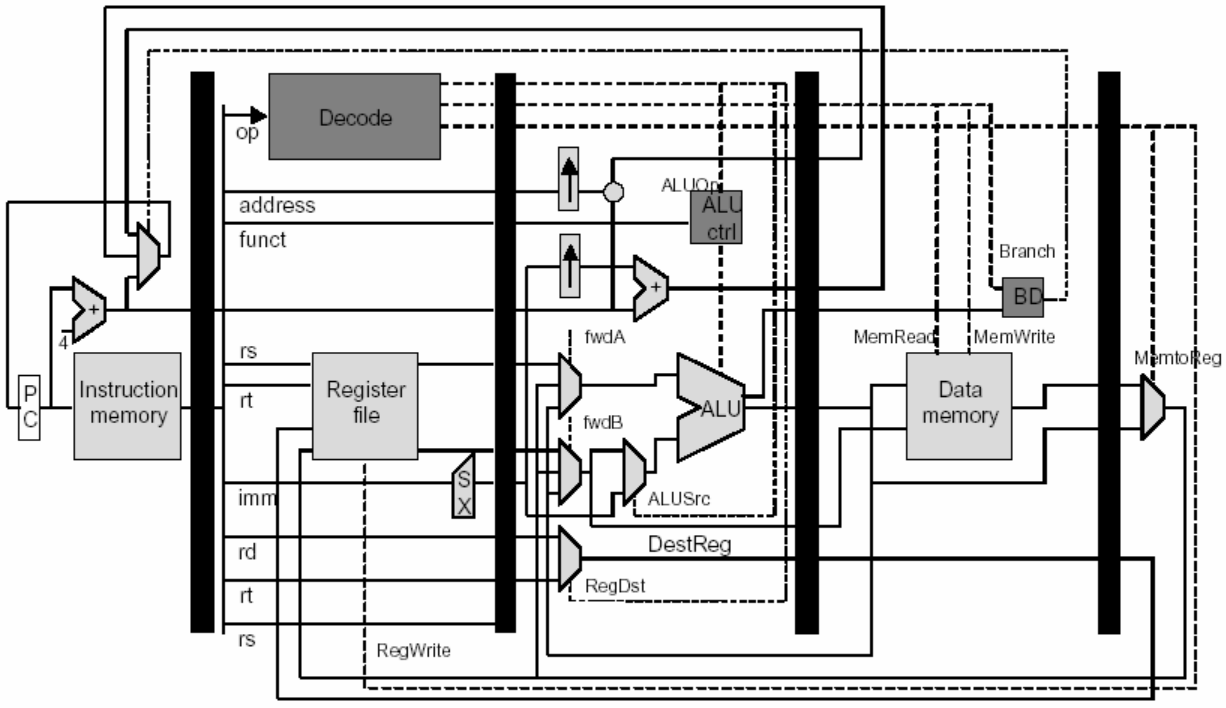
h. Ett visst program kräver 11 s CPU-tid att exekvera på MIPS med ett datacacheminne med 95% träffsannolikhet (hit rate) och 8 cyklers straffkostnad för missar (miss penalty) för alla dataåtkomster och ett instruktionscacheminne med 100% träffsannolikhet. CPI är 1, och processorns klockfrekvens är 250 MHz. Hur stor andel av instruktionerna utgjorde dataåtkomster under programexekveringen om man vet att programmet skulle kräva 9 s CPU-tid på MIPS med ett idealt minnessystem där alla minnesreferenser endast tar en klockcykel (dvs inga pipeline stalls)?
(1) 22% (X) 34% (2) 56%

i. Uttryck det decimala talet 27 på binär form i flyttalsformatet IEEE 754 med enkel precision (32 bitar). Exponenten i detta flyttalsformat är 8 bitar, och har bias 127.
(1) $01000001110110000000000000000000_2$
(X) $01100001010110000000000000000000_2$
(2) $01000001111110000000000000000000_2$

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 kan redovisning på separata papper fordras. (18 p)

a. Visa kortast möjliga sekvens av MIPS-instruktioner för att ladda konstanten 65540 i register R8. Till hjälp finns en sammanställning över MIPS maskininstruktioner i bilaga 1.

b. Nedan visas en implementering av MIPS i form av en pipeline. Styrsignalerna är inritade i bilden med streckade linjer (av utrymmesskäl så är flera styrsignaler kombinerade i en linje fram till dess de används). För multiplexrar gäller alltid att översta ingången motsvarar ett styrsignalvärde 0, ingången nedanför stysignalvärde 1, osv. Namn på styrsignalerna är angivna invid respektive styrsignal i närheten av där de används. Vissa andra signaler är också namngivna; delar av instruktionsformatet (se bilaga 1), och DestReg. En signals värde i ett visst pipelinesteg (IF, ID, EX, MEM, WB) kan refereras till genom en kombination av stegets namn och signalens namn. T.ex. står EX.MemRead för värdet på MemRead-signalen för den instruktion som för tillfället befinner sig i EX-steget.



Två multiplexrar för data forwarding är inkluderade, och styrs av signalerna fwdA respektive fwdB. Ange en logisk ekvation för något fall då en datakonflikt uppstår som inte kan lösas med data forwarding ovan. I de logiska ekvationerna kan booleska operatorer (AND, OR) och värden på namngivna signaler från olika pipelinesteg ingå. Det är också tillåtet att använda jämförelseoperatorer (t.ex. =, ≠).

c. Vad är det totala antalet bitar som TLB i uppgift 2 d) måste kunna lagra om utbytesalgoritmen för TLB är random?

d. För ett skivminne gäller att det består av 2 dubbelsidiga skivor med ett läs/skriv-huvud per yta. Antalet cylindrar är 50000 och det går 200 sektorer per spår (track). Varje sektor rymmer 512 bytes och skivorna roterar med 6000 varv per minut. Söktiden (seek time) är 10 ms. Hur stor andel av den totala accesstiden måste i medeltal ägnas åt att söka upp rätt block om 1 KB stora block accessas på slumpmässiga ställen på skivminnet om man antar att 99% av alla accesser sker till block som följer omedelbart efter det block som accessades gången innan?

e. I samband med en processkrympning från 0.25 µm till 0.18 µm har möjligheten öppnats för att lägga 256 KB L2-cache på processorchipet. Tidigare versioner av processorn har haft 2x64 KB L1-cache och 1 MB extern L2-cache. En fördel med att lägga L2-cache på processorchipet är att man får mycket snabbare accesstid till cachen. Med den gamla designen tog en access till L2-cachen 30 ns. Med den nya designen kommer en access endast att ta 5 ns. Eftersom man kommer använda samma typ av primärminne till den nya processorn så gäller det att en primärminnesaccess tar 60 ns för bägge processorerna. Man accessar inte en nivå i minneshierarkin förrän man vet att man har missat i tidigare nivå.
För ett representativt benchmark vet man vad träffsannolikheten för 1 MB cache är 80% och för 256 KB cache är träffsannolikheten 55%. Man vet också att 5 % av alla dataminnesaccesser missar i L1-datacachen och 1 % av alla instruktionshämtningar missar i L1-instruktionscachen samt att 25 % av alla instruktioner är av datatransfer typ.
Målfrekvensen för den nya processorn är 1GHz medan den gamla processorn endast kunde köras i 800 MHz. Den gamla processorn tog 20 sekunder att köra benchmarket som är helt processor/minnes begränsat (dvs. ingen tid tas upp av I/O) och består av 10 miljarder instruktioner.
Hur lång tid kommer benchmarkprogrammet ta att köra på den nya processorn?

f. Om det binära ordet $10101111100000000000000000000000_2$ tolkas som ett heltal i tvåkomplementrepresentation, vilket heltal representerar ordet? Uttryck svaret på formen $X*2^Y$.

4. 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 75 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.

   a. Hur många processorcykler tar det som mest från det att en cachemiss detekteras i CPU tills det sökta blocket laddats in om man antar att write-back tillämpas och att processor-minnebussen inte är upptage av annan aktivitet (t.ex. DMA från systembussen)? (6 p)

   b. Vad är den maximala effektiva bandbredd som kan uppnås med processor-minnebussen i föregående deluppgift? (2 p)

   c. Räkna upp de fyra typer av bussarbitrering som gåtts igenom i kursen och förklara kortfattat hur de fungerar. (4p)

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 (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