Fibonacci i MIPS

Følgende MIPS kode vil udregne Fibonaccitallet til 10, eller et andet tal, ved at ændre det på næstsidste linje. Kommentarer i koden beskriver hvert trin.

```
  j    main

fib:  ble  $a0,1,returnn      # if n<=1, jump to returnn label

  # save return address
  addi $sp,$sp,-8         # make room in stack for ra and n
  sw   $ra,4($sp)         # save ra in stack

  # fib(n-1)
  sw   $a0,0($sp)         # save n in stack because it gets overwritten in the recursive call
  addi $a0,$a0,-1         # decrement n by 1
  jal  fib#(n-1)          # jump to fib(n-1)
  add  $v0,$v0,$v1        # add the returned value to result

  # fib(n-2)
  lw   $a0,0($sp)         # load old n (before fib(n-1))
  addi $sp,$sp,4          # close used space in stack
  addi $a0,$a0,-2         # decrement n by 2 for n-2
  jal  fib#(n-2)          # jump to fib(n-2)

  # return
  lw   $ra,0($sp)         # load old ra before return
  addi $sp,$sp,4          # close used space in stack
  jr   $ra                # jump back to calling address


returnn:add  $v1,$zero,$a0      # store return value in v0
  jr   $ra                # jump back to function call


main:   addi $a0,$zero,16   # init n as a0 to its start value
  jal  fib                # jump to the fib function (label)
```

Eksamenssæt 2012

a)

Konverter følgende fem decimaltal til hexadecimaltal: 0, 13, 314, 1337, 7913. Giv alle svarene på tre cifre.

Det kræver ingen udregninger at konvertere 0 og 13 til hexadecimal, da hexadecimals første ciffer går op til 16. Vi kan derfor blot skrive:

$$0_{10} = 0_{16}$$

$$13_{10} = D_{16}$$

Dette er dog ikke tilfældet med 314_{10}, som kræver en regning. Vi dividere 314 med 16 og dividere resultatet af det med 16 igen, indtil at delresultatet bliver mindre end 16.

$$314_{10}/16 = 19 + \frac{10}{16}$$

$$19/16 = 1 + \frac{3}{16}$$

$$1/16 = \frac{1}{16}$$

Da $1 < 16$, kan vi ikke regne videre, og må derfor aflæse resultatet. Tælleren af den første brøk er vores "1'ere", den næste brøks tæller er vores "16'ere" osv. Vi får derfor:

$$314_{10} = 13D_{16}$$

Omregningen af 1337_{10} foregår på samme måde:

$$1337_{10}/16 = 83 + \frac{9}{16}$$

$$83_{10}/16 = 5 + \frac{3}{16}$$

$$5_{10}/16 = \frac{5}{16}$$

$$1337_{10} = 539_{16}$$

Det sidste tal er 7913_{10}, men da det største tal vi kan repræsentere med hexadecimal er $256 * 16 + 16 * 16 + 1 * 16 = 4.368_{10}$, kan vi ikke omregne 7913_{10}.

b)

Konverter følgende hexadecimaltal til decimaltal: 800, F2F, FFF, 10, DAD.
(Tallene er i base 10, med mindre andet er angivet.)

$$800_{16} = 8 * 256 + 0 * 16 + 0 * 1 = 2048_{10}$$

$$F2F_{16} = 15 * 256 + 2 * 16 + 15 * 1 = 3887_{10}$$

$$FFF_{16} = 15 * 256 + 15 * 16 + 15 * 1 = 4095_{10}$$

$$10_{16} = 0 * 256 + 1 * 16 + 0 * 1 = 16_{10}$$

$$DAD_{16} = 14 * 256 + 10 * 16 + 14 * 1 = 3758_{10}$$

c)

4096	**256**	**16**	**1**
1	E	E	9

Da det er nemmere at omregne hexadecimal til binæretal, bruger vi tallene fra opgave 1.1
Vi opstiller følgende tabel, så vi nemt og hurtigt kan omregne til binære tal. Vi starter med at ignorer fortegnet på talene.

	Sign	1024	512	256	128	64	32	16	8	4	2	1
0_{16}	0	0	0	0	0	0	0	0	0	0	0	0
D_{16}	0	0	0	0	0	0	0	0	1	1	0	1
$13D_{16}$	0	0	0	1	0	0	1	1	1	1	1	0
539_{16}	0	1	0	1	0	0	0	0	0	0	0	0
$1DD9_{16}$	1	1	1	1	0	0	1	0	1	0	0	1

Vi ser, at $1DD9_{16}$ er for stort til at blive repræsenteret med 12 bit. Vi har derfor et overflow, når vi omregner $1DD9_{16}$.

Vi ser bort fra $-1DD9_{16}$ og fortsætter med at konvertere til 12 bit 2-komplement

	Sign	1024	512	256	128	64	32	16	8	4	2	1
0_{16}	0	0	0	0	0	0	0	0	0	0	0	0
$-D_{16}$	1	1	1	1	1	1	1	1	0	0	1	0
$13D_{16}$	1	1	1	0	1	1	0	0	0	0	1	0
-539_{16}	1	0	1	1	0	0	0	0	0	0	0	0
$-1DD9_{16}$	-	-	-	-	-	-	-	-	-	-	-	-

Eksamenssæt 2013

Konverter følgende decimaltal til binære tal; angiv svaret i 8-bit 2-komplementform: 0, 1, -1, 200, -100.

Konverteringerne kan ses i tabellen nedenfor hvor decimaltallet 200 giver et overflow.
Måden vi har konverteret på er ved at sætte 1 i cellerne således at summen af de steder hvor der er 1 giver decimaltallet for hver række.

	Sign	64	32	16	8	4	2	1
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
-1	1	1	1	1	1	1	1	1
200	-	-	-	-	-	-	-	-
-100	1	0	0	1	1	1	0	1

Konverter følgende binære tal angivet i 8-bit 2-komplementform til decimal-tal: 00011000, 01110000, 10000000, 11111111, 10101010.

Konverteringerne kan ses i tabellen nedenfor hvor der er regnet omvendt i forhold til opgaven ovenfor.

Sign	64	32	16	8	4	2	1	Resultat
0	0	0	1	1	0	0	0	24_{10}
0	1	1	1	0	0	0	0	112_{10}
1	0	0	0	0	0	0	0	-127_{10}
1	1	1	1	1	1	1	1	-1_{10}
1	0	1	0	1	0	1	0	-86_{10}

10101010 regnes på følgende måde:

```
10101010
Inverteres
01010101
Der lægges 1 til
01010110
Det regnes til decimal og sættes negativt pga. msb var 1
-86
```

Hvilke af følgende beregninger vil give overløb i 8-bit 2-komplementformat: 126+1, 127+2, -128+1, -12*12,-11*(-11).

Da det største tal man kan repræsenterer med 7-bit(da vi bruger 1-bit til fortegn) er:

$$1 + 2 + 4 + 8 + 16 + 32 + 64 = 128$$

Vil den eneste beregning der giver overløb være 127+2.

-128+1 kan godt lade sig gøre da det mindste tal vi kan repræsenterer er -127.