Tentamen
EDAA05 – Datorer i system

2014–10–27, 14.00–19 .00

Tillåtna hjälpmedel: *bifogad formel- och symbolsamling*.

För godkänt betyg på tentamen krävs minst 20 poäng av totalt 30 möjliga.

Börja med att fylla i personuppgifter och övrig om kurs/datum på det utdelade tentamensomslaget. Ha legitimation i beredskap för identitetskontroll. Påbörja därefter tentamen. Skriv dina svar på separat papper. Det går bra att skriva lösningarna till flera uppgifter på samma papper. Märk varje papper med dina initialer i överkanten. När du är klar lämnar du in dina lösningar i det utdelade tentamensomslaget. Behåll gärna detta papper med uppgifterna om du vill.

Lycka till!

---

1. Moores lag

   a) Redogör kortfattat (med en eller några få meningar) för vad Moores lag innebär.

   (1p)

   b) Ange något skäl till att vi inte kan förvänta oss att Moores lag ska fortsätta att gälla i framtiden.

   (1p)

2. Olika talbaser och aritmetik.

   a) Man kan ibland tala om ett binärt tals *minst signifikanta* respektive *mest signifikanta* siffra. Förklara dessa båda begrepp kortfattat, t.ex. genom ett exempel.

   (2p)

   b) Skriv det decimala talet 267 som ett binärt tal och därefter som ett hexadecimalt tal.

   (2p)

   c) Skriv det decimala talet -106 som ett 8-bitars binärt tal i tvåkomplementsform.

   (2p)

   d) Skriv det hexadecimala talet CE i decimalform.

   (1p)

   e) Utför *subtraktionen* $11011100 - 1110011$. Svara genom att ställa upp talen på formen

   ```
     xxxx
   - yyyy
   ------
   ```

   och utför subtraktionen så att svaret framgår tillsammans med vilka lån som gjorts.

   (2p)

3. Teckenkodning

   a) Vilka av nedanstående fyra påståenden är sanna?

      1) Med hjälp av UTF-8 kan vi representera lika många olika tecken som UTF-16 och UTF-32.
      2) Om vi vet att en textfil är kodad enligt UTF-8 vet vi också att radsluten är representerade med ett enkelt tecken med teckenkoden 10 (hexadecimalt 0A, ””, ”linefeed”).
      3) ISO 8859 använder alltid lika många bitar för att representera ett tecken.
      4) Det finns alltid en s.k. BOM (Byte Order Mark) i början på alla textfiler kodade i enlighet med Unicode-standarden.

   (2p)

   b) Beskriv kortfattat – med några få meningar – hur Unix-kommandot "`od`"kan användas för att ta reda på hur en text är kodad.

   (1p)

   c) Hur brukar datorer som kör operativsystemet Unix (eller Linux) vanligtvis representera radslut i en textfil?

   (1p)

4. Boolesk algebra

   a) Logik-Lars vill kontrollera att han minns hur man använder De Morgans lagar för att skriva om uttrycket $\neg(x \vee y \wedge z)$. Genom att flytta in ”icke” i parentesen och byta mellan ”och” och ”eller” får han då uttrycket $\neg x \wedge \neg y \vee \neg z$. Använd sanningstabeller för att avgöra om likhet gäller mellan uttrycken. Svara genom att redovisa dina sanningstabeller samt motivera ditt svar utgående från dem.

   (2p)

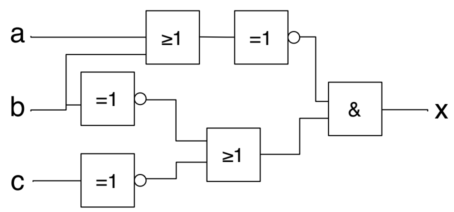
   b) Förenkla nedanstående booleska uttryck så långt det går med hjälp av räknelagarna i den bifogade formelsamlingen. Redovisa varje delsteg i förenklingen så att det framgår vilken räknelag som använts (du behöver *inte* skriva ut namnen på lagarna).

   $\neg(q \wedge p) \vee p$

   (2p)

5. Booleska uttryck och digitala grindar

   a) Skriv ett booleskt uttryck för signalen x i figuren till höger. Svara med en funktion av insignalerna a, b och c. Uttrycket ska ej förenklas.

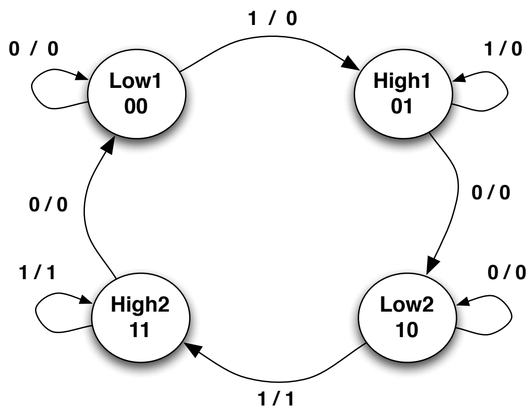


   (2p)

   b) Rita med hjälp av grindsymboler (se den bifogade grindsymbolsamlingen) upp ett kombinatoriskt nät som realiserar det booleska uttrycket $\neg(a \vee b) \wedge c \vee \neg d$ (förenkla inte uttrycket).

   (2p)

6. Realisering av tillståndsmaskin
   Roger och Jonas har designat en tillståndsmaskin enligt figuren nedan. Tillståndsmaskinen har en insignal, *i* och en utsignal, *o*. Roger och Jonas har dessutom numrerat tillstånden från 00 till 11 (binärt). Avsikten med tillståndsmaskinen är att *varannan* gång insignalen går från 0 till 1 ska utsignalen bli 1. Den ska sedan förbli 1 tills insignalen blir 0 igen.



Nedanstående sanningstabell beskriver hur utsignalen och tillståndsövergångarna ska ske i tillståndsmaskinen ovan. Givet insignalen, *i*, och det nuvarande tillståndet, representerat av *s1* och *s2*, visas vad utsignalen, *o* och det nya tillståndet (*s1′* och *s2′*) ska bli.

i	s1	s2	o	s1′	s2′
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	0	1
1	1	0	1	1	1
1	1	1	1	1	1

a) Hjälp Roger och Jonas att ta fram booleska uttryck för utsignalen, *o*, samt de nya tillstånden, *s1′* och *s2′*. Dvs, skriv booleska funktioner (uttryckta i *i*, *s1* och *s2*) enligt:
   $o = ...$
   $s1' = ...$
   $s2' = ...$

(3p)

b) Hjälp Roger och Jonas att rita upp hur den färdiga tillståndsmaskinen ska realiseras med hjälp av digitala grindar. Förutom grindsymbolerna i formelsamlingen kan följande två symboler vara användbara:

(4p)

Slut!

Formelsamling och grindsymbolförteckning

$x \vee 0 = x$
$x \wedge 1 = x$

$x \vee 1 = 1$
$x \wedge 0 = 0$

$x \wedge \neg x = 0$
$x \vee \neg x = 1$

$x \vee y = y \vee x$
$x \wedge y = y \wedge x$

$x \vee (y \vee z) = (x \vee y) \vee z$
$x \wedge (y \wedge z) = (x \wedge y) \wedge z$

$x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

$\neg(x \vee y) = \neg x \wedge \neg y$
$\neg(x \wedge y) = \neg x \vee \neg y$