Digital- och datorteknik - har du uppnått kursmålen?

exempel på Tentamen 1

Fredag xx januari 2004, kl 08.30 - 12.30 i vv-salar

Examinator
Stig-Göran Larsson, tel. 772 1693

Kontaktperson under tentamen
Stig-Göran Larsson, tel. 772 1693

Tillåtna hjälpmedel

Häftet

"Instruktionslista för FLEX"

"Instruktionslista för CPU12"

I den får rättelser och understrykningar vara införda, inget annat.

Tabellverk och miniräknare får ej användas!

Allmänt
Siffror inom parentes anger maximal poäng på uppgiften. Maximal poäng kan fås om:

- redovisningen av svar och lösningar är läslig och tydlig. **OBS!** Ett lösningsblad får endast innehålla redovisningsdelar som hör ihop med en uppgift.
- din lösning ej är onödigt komplicerad.
- du motiverat dina val och ställningstaganden
- redovisningen av en hårdvarukonstruktion innehåller funktionsbeskrivning, lösning och realisering.
- redovisningen av en mjukvarukonstruktion är fullständigt dokumenterad, d v s är redovisad både i strukturform (flödesplan eller pseudospråk) och med kommenterat program i assemblerspråk, om inget annat anges i uppgiften.

Betygsättning
För godkänt slutbetyg på kursen fordras att både tentamen och laborationer är godkända. Poäng på tentamen bestämmer slutbetyget enligt skalan

20p ≤ **betyg 3** < 30p ≤ **betyg 4** < 40p ≤ **betyg 5**

Lösningar
anslås på kursens hemsida tidigast kl 09.00 dagen efter tentamen.

Betygslistan
anslås såsom anges på kursens hemsida.

Granskning
Tid och plats anges på kursens hemsida.

Institutionen för datorteknik
CHALMERS TEKNISKA HÖGSKOLA

Upg 1 *Talomvandling, koder, aritmetik och flaggor*.

I uppgift a-d nedan används 5-bitars tal. X = 11100 och Y = 00101.

a) Visa med penna och papper hur räkneoperationen R = X-Y utförs i en dator (i en ALU). **(1p)**

b) Ange sedan flaggbitarna N, Z, V, C. **(1p)**

c) Tolka bitmönstren R, X och Y som tal *utan* tecken och ange dess decimala motsvarighet. **(1p)**

d) Tolka bitmönstren R, X och Y som tal *med* tecken och ange dess decimala motsvarighet. **(1p)**

e) Studera bitmönstren 11011100_2 och 01011010_2. Kan bitmönstren representera följande:
1) ett negativt tal på tecken-beloppsform
2) ett 7-bitars ord utökat med en jämn paritetsbit (Ge ditt svar i tabellform enligt:)
3) Gray-kod
4) NBCD-tal

(2p)

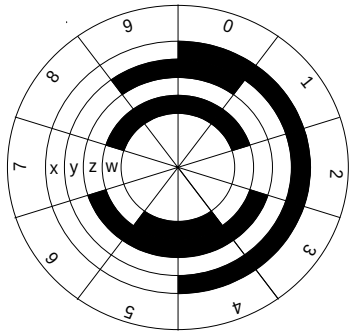
	11011100	01011010
1	Ja/Nej	Ja/Nej
2		
osv…		

f) I figuren till höger visas den kodskiva som du arbetat med vid laboration 1. Antag att den skall användas till ett digitalt "chokladhjul". Fundera lite över dess uppbyggnad.

Kodorden **xyzw** är kodade med Excess-3 Graykod.

Svart fält = 0 och genomskinligt fält = 1.

Vilka kännetecken ser du på skivan på att det å ena sidan är en **Graykod** och å andra sidan är en **Excess-3 kod** som används? **(2p)**



Upg 2 *Digitalteknik - småfrågor*

a) Man behöver en 2-ingångs AND-grind men har bara 2- ingångs OR-grindar och 2-ingångs XOR-grindar. Hur kopplar man upp AND-grinden med dessa? **(2p)**

b) Funktionen f(x,y,z) = yz + xz' + x'y'z är given.
Ange funktionen på konjunktiv normal form och konjunktiv minimal form. **(2p)**

c) Rita symbolen för en "1 av 4 väljare" (en MUX). Ange även väljarens funktionstabell. **(2p)**

Upg 3 *Digitalteknik - konstruktion*

a) Konstruera och rita upp en räknare som har räknesekvensen **1-3-5-7**-1-3-5-7.....
Använd T-vippor med positiv flanktriggning och vanliga grindar (AND, OR, NOT).
Konstruktionen skall vara minimal. Du kan bortse från hur räknaren startas. **(5p)**

b) Funktionsbeskrivningen är känd genom funktionstabellen till höger. Den beskriver en Boolesk funktion f(x,y,z,w). "-" avser "don't care"-termer.

Skriv funktionen på minimal disjunktiv form och realisera den med NAND-grindar med valfritt antal ingångar samt INVERTERARE. **(3p)**

w	x	y	z	F
0	0	0	0	0
0	0	0	1	-
0	0	1	0	-
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	-
1	1	0	1	1
1	1	1	0	1
1	1	1	1	-

c) Realisera funktionen i uppgift 3b med så få valfria (AND, OR, XOR, NAND, NOR) tvåingångars grindar som möjligt samt INVERTERARE.
(6p - antal grindar du använder)

Upg 4 *Dataväg och styrenhet för FLEX.*

I Appendix visas hur datorn FLEX är uppbyggd och hur ALU:ns funktion väljs med styrsignalerna f_3 - f_0 och C_{in}. Ett exempel på mikroprogram och styrenhetens principkoppling visas också.

a) I tabellen nedan visas RTN-beskrivningen för EXECUTE-sekvensen för en instruktion för FLEX-processorn. NF i tabellens sista rad anger att nästa tillstånd (state) skall vara det första i FETCH-sekvensen. Rita en tabell där du anger State nr (0..3) och Styrsignaler. Endast styrsignaler = 1 skall anges. Du kan utelämna RTN-beskrivningen i din tabell. **(1p)**

State nr	RTN-beskrivning	Styrsignaler (=1)
0	PC→MA, PC+1→PC	
1	M→MA,	
2	M→T	
3	A-T, Flaggor →CC, NF	

b) Förklara med ord vad instruktionen ovan utför i varje klockcykel.
Ange sedan instruktionen med assemblerspråk för FLEX-processorn (Ex: `LDA Adr`). **(2p)**

c) Rita en tabell motsvarande den ovan, som visar utförandefasen för maskininstruktionen **JSR Adr** för FLEX-processorn.
Instruktionen beskrivs i instruktionslistan för FLEX-processorn. **(4p)**

Mikroprogrammering av FLEX!
FLEX skall utrustas med en ny instruktion **WFP #Data,Adr**
Maskininstruktionen består av tre byte, nämligen OP-kod, Data och Adress.
Instruktionen jämför hela tiden **Data** med minnesinnehållet på adressen **Adr** och när dessa är lika är instruktionen färdig och en ny FETCH anropas.

Maskininstruktionen har OP-koden $E3. Ledigt mikrominne är i adressområdet [$500,$7FF].

Ledning/Tips/Begränsningar:
CC-registret har ett odefinierat värde när instruktionen är färdig.
Register A, B, X och S skall vara oförändrade. **(5p)**

Sekvens	Adress (Hex)	Hopp villkor G_K	Hopp adress (Hex)	Styrsignaler (aktiva)	RTN

Upg 5 *Småfrågor rörande FLEX-processorn*

a) Vad menas med ett assemblerdirektiv respektive en assemblerinstruktion? **(1p)**

b) Raden nedan är hämtat från ett program.

```
BRA  Stop
```

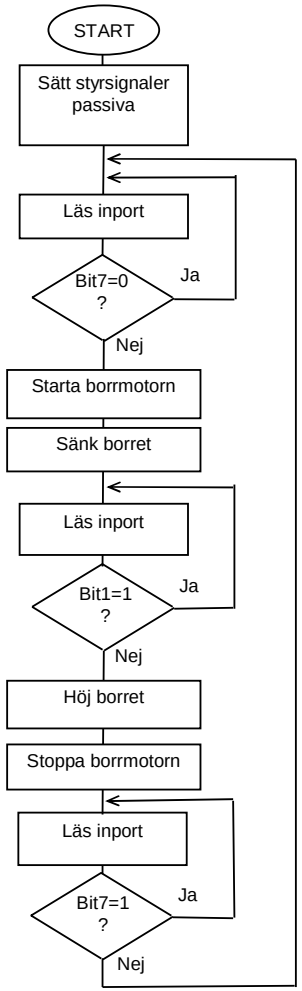
Operationskoden för BRA-instruktionen är placerad på adress $CD i minnet och symbolen Stop finns på adress $BE. Översätt instruktionen till maskinkod och ange denna. **(2p)**

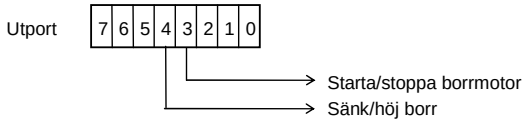
c) Rita en bild av aktuellt minnesinnehåll efter att kodsekvensen nedan assemblerats och laddats till labbsystemet. Ange adresser, data och maskinkod.

```
      ORG  $C0
T1    EQU  17
T2    EQU  $3A
T3    RMB  3
T4    FCB  3,$A,%11
      NOP
      LDA  128
      LDB  T1,X
      STB  T3
      EORB #T2
      INCB
      - fortsättning -
```
(3p)

Upg 6 ***Skriv program för FLEX-processorn***

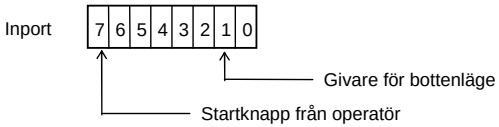
I labbet styrde du borrmaskinen med FLEX-datorn. Skriv ett assemblerprogram enligt flödesplanen nedan. Programmet skall placeras i minnet med start på adress $40.
Skriv ett väldokumenterat program innehållande enkla radkommentarer där du utnyttjar ”EQU-satser” för definitioner av konstanter och portar. Ut- och In-portarna är placerade på adress $FD respektive $FE. Observera att varje box i flödesplanen upptar en eller flera instruktioner **(7p)**





Av figuren ovan och beskrivningen nedan framgår att bitarna 3 och 4 på utporten styr var sin funktion hos borrmaskinen.

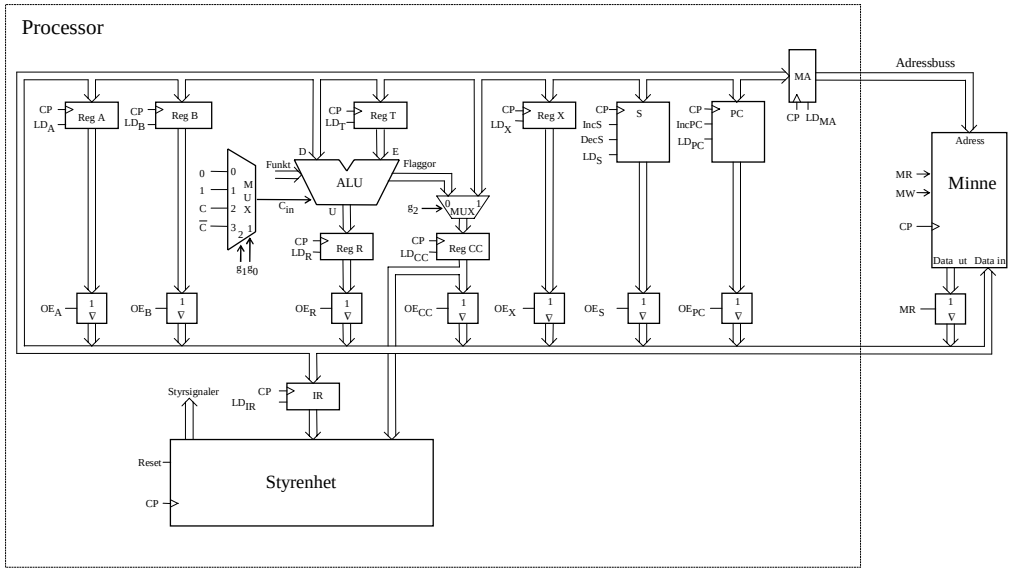
- **Starta borrmotor:**
  Borrmotorn startas genom att "0" matas ut på bit 3 på utporten.
- **Stoppa borrmotor:**
  Borrmotorn stoppas genom att "1" matas ut på bit 3 på utporten.
- **Sänk borr:**
  Borret sänks genom att "0" matas ut på bit 4 på utporten.
- **Höj borr:**
  Borret höjs genom att "1" matas ut på bit 4 på utporten.



Av figuren ovan och punkterna nedan framgår hur insignalen från borrmaskinen fungerar.

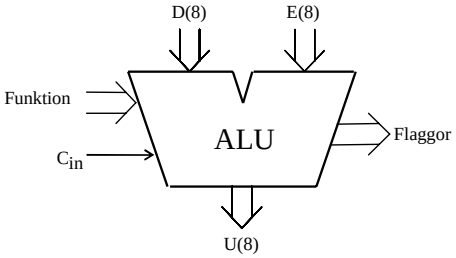
- **Borr i bottenläge:**
  Givarvärde "0".
- **Borr ej i bottenläge:**
  Givarvärde "1".
- **Operatören trycker på startknappen**
  Bit 7 = 1
- **Operatören trycker inte på startknappen**
  Bit 7 = 0

FLEX-datorn Bilaga 1



ALU:ns funktion

Bilaga 2



ALU:ns **operation (logik- eller aritmetik-)** på indata **D**, **E** och $\mathbf{C_{in}}$ bestäms av insignalerna **Funktion [F = (f_3, f_2, f_1, f_0)]** enligt tabellen nedan.. I kolumnen Operation förklaras, när det behövs, hur operationen utförs. Med"+" och "–" avses **aritmetiska operationer**.

f_3 f_2 f_1 f_0	U = f(D,E,C_{in})	
	Operation	**Resultat**
0 0 0 0	bitvis nollställning	0
0 0 0 1		D
0 0 1 0		E
0 0 1 1	bitvis invertering	D_{1k}
0 1 0 0	bitvis invertering	E_{1k}
0 1 0 1	bitvis OR	D OR E
0 1 1 0	bitvis AND	D AND E
0 1 1 1	bitvis XOR	D XOR E
1 0 0 0	D + 0 + C_{in}	D + C_{in}
1 0 0 1	D + FFH + C_{in}	D – 1 + C_{in}
1 0 1 0		D + E + C_{in}
1 0 1 1	D + D + C_{in}	2D + C_{in}
1 1 0 0	D +E_{1k} + C_{in}	D – E – 1 + C_{in}
1 1 0 1		0
1 1 1 0		0
1 1 1 1	bitvis ettställning	FFH

Flaggorna är utsignaler och för de gäller:

Carryflaggan (C) är minnessiffran ut (carry-out) från den mest signifikanta bitpositionen (längst till vänster) när en aritmetisk operation utförs av ALU:n.

Vid **subtraktion** gäller för denna ALU att
C = 1 om lånesiffra (borrow) uppstår och
C = 0 om lånesiffra inte uppstår.

Carryflaggans värde är 0 vid andra operationer än aritmetiska.

Overflowflaggan (V) visar när en aritmetisk operation ger "overflow" enligt reglerna för 2-komplementaritmetik.

V-flaggans värde är 0 vid andra operationer än aritmetiska.

Zeroflaggan (Z) visar när en ALU-operation ger värdet noll som resultat på U-utgången.

Signflaggan (N) är identisk med den mest signifikanta biten (teckenbiten) av utsignalen U från ALU:n.

Half-carryflaggan (H) är minnessiffran (carry) mellan de fyra minst signifikanta och de fyra mest signifikanta bitarna i ALU:n.

H-flaggans värde är 0 vid andra operationer än aritmetiska.