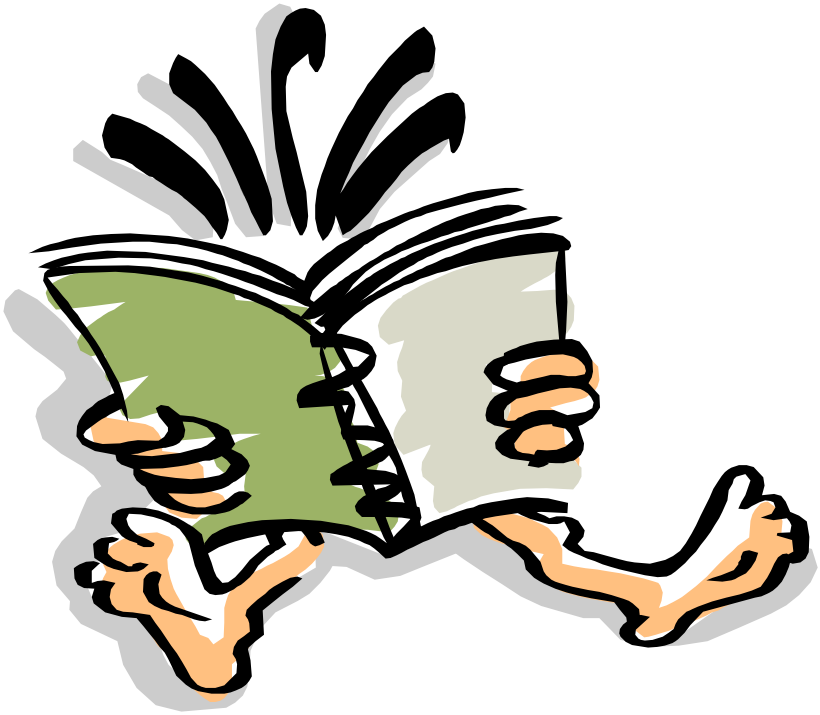


Tentamen
Mikrodatorteknik, 4p/5p

2001-08-22

Skrivtid:	5 timmar
Preliminära gränser 5p:	G: 52-79, VG: 80- (Max: 100p)
Preliminära gränser 4p:	G: 48-79, VG: 80- (Max: 100p)
Ansvariglärare:	Mattias O'Nils
Telefon:	070-695 7668
Hjälpmedel:	The 68000's Instruction Set Miniräknare
Program:	Systemteknikprogrammet, ST3

OBS!
Skriv tydligt!
Skriv namn och kod på varje blad.
Skriv bara på en sida av varje blad.
Redovisa alla steg i beräkningar och konstruktioner.
Rita snygga figurer!

Lycka till,
/Mattias

1. Beskriv skillnaden mellan Harward och Von Neumann-arkitekturen. **6p**

2. Hur kan man implementera loopen

```
while(a<5){
  kalle();
  a++;
}
```

i assemblerkod. Där `kalle` är en subrutin och `a` är ett 16-bits tal i minnet. **6p**

3. Vilka två instruktioner används vid ovillkorliga hopp i 68000 processorn och vad är skillnaden på dessa två?

6p

4. Vad skiljer RESET från övriga undantag i 68000 processorn (nämn minst 2 saker). **8p**

5. Beskriv med ett exempel *direct mapping* adressering av ett cacheminne. **8p**

6. Antag att register D1 innehåller $000000A8. Ange D1, C-flaggans samt Z- flaggans värde efter instruktionerna nedan har exekverats.

```
a. ADDI.B    #11,D1
b. SUBI.B    #$70,D1
c. ADDI.B    #$A0,D1
d. SUBI.B    #$A8,D1
e. ASR.B     #6,D1
f. LSL.B     #3,D1
```

6p

7. Nämn två sätt att synkronisera I/O operationer samt beskriv dessa. **6p**

8. Adresseringsmetoder: Vad kommer D2 och A3 att innehålla efter att instruktionen har körts. Innehållet i minnet är specificerat av tabellen på nästa sida. Instruktionerna är oberoende av varandra och D2 är $00000000 samt A3 = $0000F000 före varje instruktion exekveras.

```
a)  MOVE.L #$EFFC, D2
b)  MOVE.W  -2(A3), D2
c)  MOVE.L -(A3), D2
d)  MOVE.B -(A3), D2
e)  MOVE.W 4(A3), D2
f)  MOVE.B $EFFC, D2
g)  MOVE.L (A3)+,D2
h)  MOVE.W (A3)+,D2
```

8p

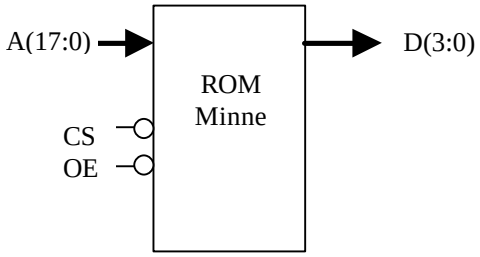
Adress (Hexadecimal)	Innehåll (Hexadecimal)
EFFC	77
EFFD	21
EFFE	C3
EFFF	FD
F000	00
F001	14
F002	99
F003	74
F004	A6
F005	AA

9. Programmet nedan består av 16 stycken instruktioner, din uppgift är att tala om vilka instruktioner som kommer att exekveras och varför. (Det är inget fel på programmet, ni ska bara fundera ut vilka instruktioner som exekveras och varför)

```
            ORG         $11000                  * Instruktions nummer:
            MOVE.B      #$F0,D0                 * 1
            ASR.B       #4,D0                   * 2
            BPL         Label_1                 * 3
            ANDI.B      #$6F,D0                 * 4
Label_1     EORI.B      #$6F,D0                 * 5
            BEQ         Label_2                 * 6
            EORI.B      #$60,D0                 * 7
            BEQ         Label_3                 * 8
Label_2     SUB.B       #1,D0                   * 9
            ANDI.B      #$80,D0                 * 10
            BMI         Label_4                 * 11
Label_3     ORI.B       #$7F,D0                 * 12
            BEQ         Label_4                 * 13
            ANDI.B      #$80,D0                 * 14
            BEQ         Label_4                 * 15
Label_4     BRA         Label_4                 * 16
```

10p

10. Sättsamman ett ROM-minne på 2 Mbyte (16-bitar bred), med hjälp av minneskomponenter 256kx4bit (se komponenten). Ni för gärna använda en färdig avkodare för att generera CS-signalerna.
Obs! Generera bara CS signalen. **10p**



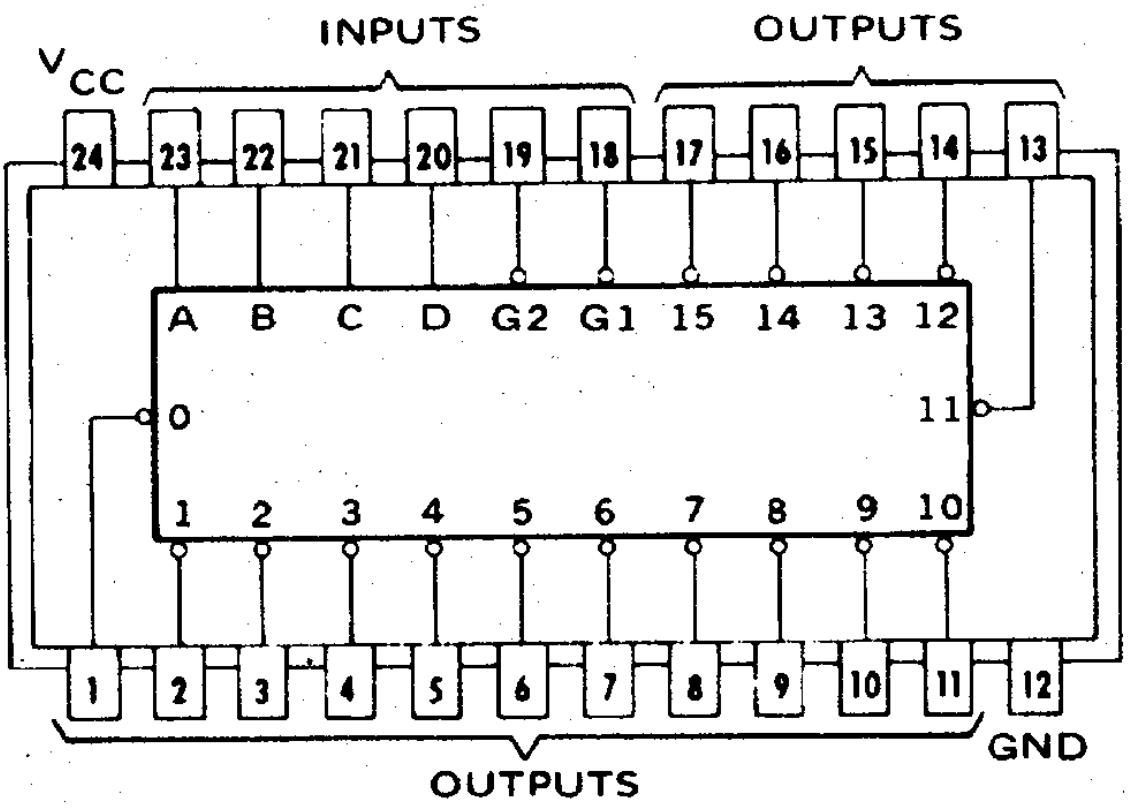
11. Konstruera en partiell minnesavkodare med följand minneskarta. Använd en eller flera 4-16 avkodaren 74HC154 beskriven på nästa sida. (Dvs. generera CS signaler för 68000)

12p

ROM1	00 0000 -	03 FFFF
ROM2	04 0000 -	04 FFFF
RAM1	05 0000 -	05 FFFF
RAM2	06 0000 -	07 FFFF
IO_0	08 0000 -	08 0FFF
IO_1	08 1000 -	08 1FFF
IO_2	08 2000 -	08 3FFF
IO_3	08 4000 -	08 7FFF
IO_4	08 8000 -	08 FFFF

12. Skriv ett program som räknar antal knapptryckningar. Knappen är ansluten till bit 3 i IO_REG (IO_REG är en minnesposition på 8-bitar med adress $F000). När knappen är nedtryckt är bit 3 en etta och när den inte är nedtryckt är bit 3 en nolla. Antalet knapptryckningar skrivs till minnespositionen DISPLAY (DISPLAY är en minnesposition på 8-bitar med adress $F001). Glöm inte att deklarera IO_REG och DISPLAY! Räknaren implementeras som en Modulo-256 räknare.

14p


INPUTS
OUTPUTS
VCC
24
23
22
21
20
19
18
17
16
15
14
13
A
B
C
D
G2
G1
15
14
13
12
0
11
1
2
3
4
5
6
7
8
9
10
1
2
3
4
5
6
7
8
9
10
11
12
OUTPUTS
GND