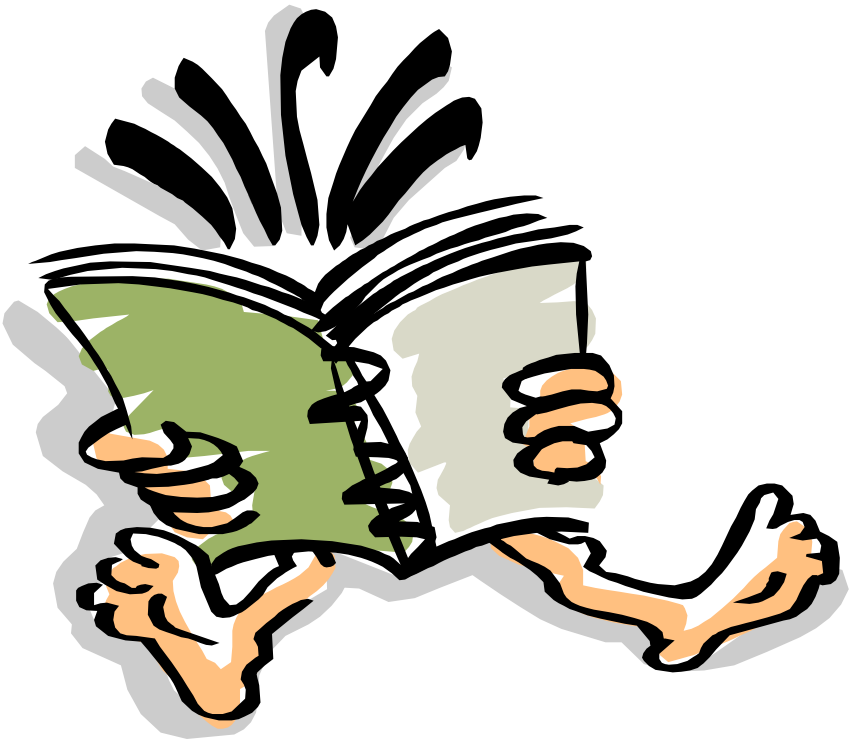


Tentamen Mikrodatorteknik, 4p

2000-01-10

Skrivtid:	5 timmar
Ansvariglärare:	Mattias O'Nils
Telefon:	070-695 7668
Hälpmedel:	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

Del A (50p)

1. Vad används programräknaren (PC) till och när förändras innehållet i denna (minst 2 exempel) **4p**

2. Beskriv hur stacken i 68000 fungerar, samt hur man läser från och skriver till denna? **5p**

3. a) Beskriv von Neumann arkitekturen, vad som karakteriserar denna och beskriv instruktionsexekveringen. **6p**

3. b) Vad kan göras för att förbättra von Neumann arkitekturen? Ge exempel på minst 2 olika tekniker, samt beskriv hur dessa ökar prestanda hos von Neumann arkitekturen. **4p**

4. Vilka tre typer av signaler består 68000-processorns buss av samt beskriv kort vad dessa har för funktion. **6p**

5. Nämn tre olika metoder att överföra parametrar till en funktion, samt för- och nackdelar med dessa. **6p**

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

```
a. ADDI.B   #11,D1
b. SUBI.B   #$70,D1
c. ADDI.B   #$A0,D1
d. SUBI.B   #$64,D1
e. ASL.B    #6,D1
f. LSR.B    #2,D1
```

7. Vad betyder RISC och CISC. Beskriv vad som är karakteristiskt för dessa olika arkitekturer. **4p**

8. a) Beskriv vad som händer vid undantag (68000). **5p**

8. b) Hur avslutas en undantagsrutin? **1p**

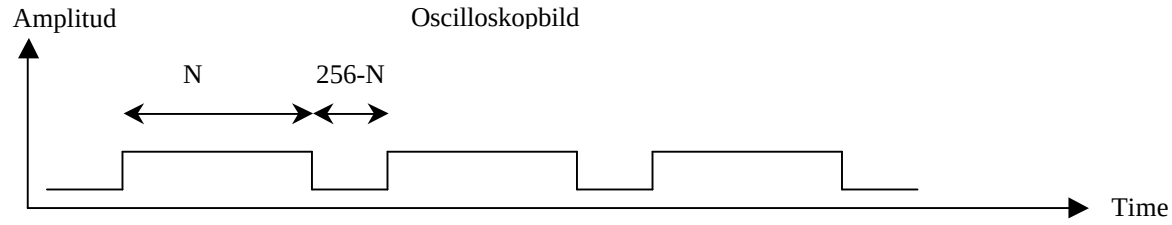
8. c) Ge minst 3 exempel på undantag i 68000. **3p**

Del B (50p)

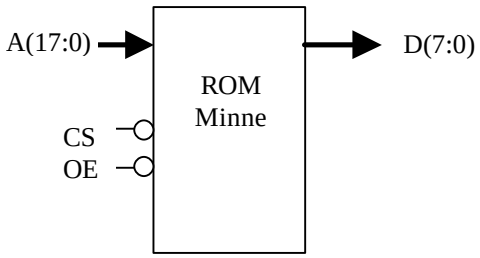
9. Implementera följande FOR-loop i 68000-assembler. Alla variabler är av 16-bit storlek och kan ersättas med register. **5p**

```
for (i=0; i<20; i++) {
    Fn = Fn_1 + Fn_2;
    Fn_2 = Fn_1;
    Fn_1 = Fn;
}
```

10. Skriv ett program som genererar en PWM (Pulsbreddsmodulerad) signal på utporten OUT_1(bit 7). Sätt OUT_1(bit 7) hög N ggr och låg 256-N ggr. Talet N (som moduleras) läses från inporten IN_1. Ett nytt N ska läsas in före varje period. (OUT_1 har adressen $FF0000 och IN_1 har adressen $FF0002) **12p**



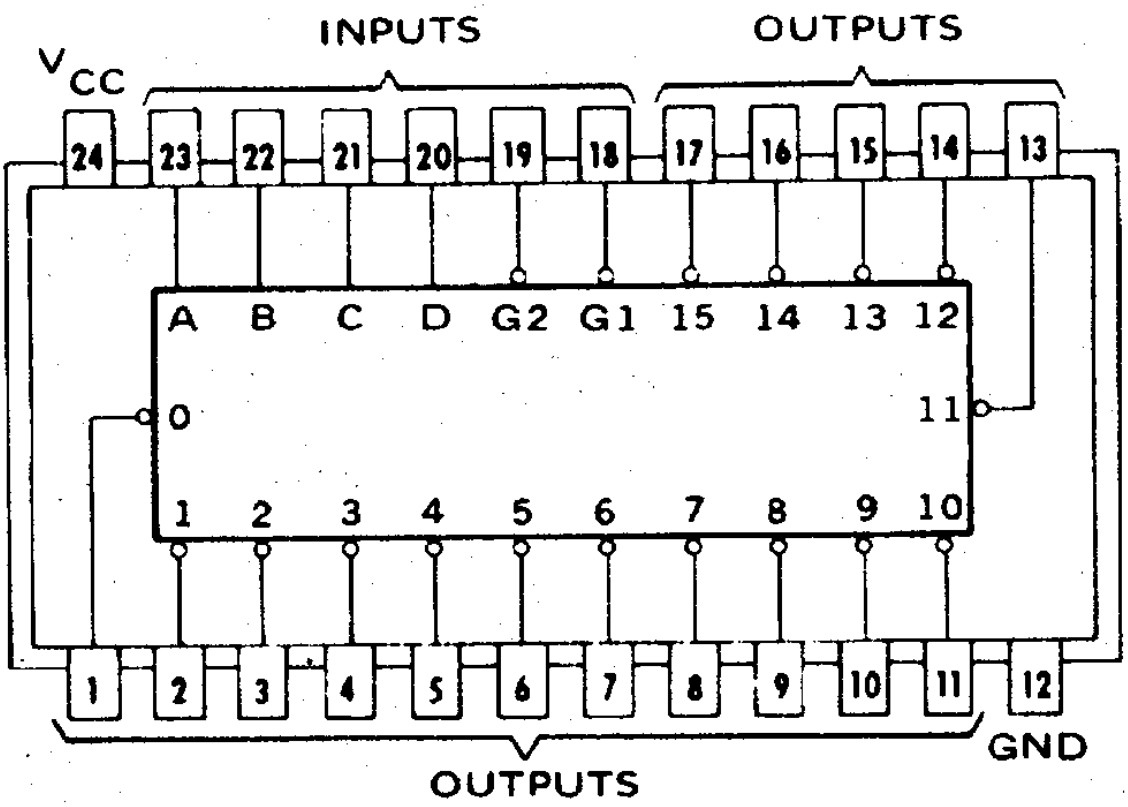
11. Sättsamman ett ROM-minne på 1Mbyte (16-bitar bred), med hjälp av minneskomponenter 256kx8bit (se komponenten). **12p**



12. Konstruera en fullständig minnesavkodare (för 68000) med följand minneskarta. Använd 4-16 avkodaren 74HC154 beskriven på nästa sida. (Dvs. generera CS signaler) **9p**

ROM	00 0000 -	01 FFFF
RAM1	02 0000 -	04 FFFF
RAM2	05 0000 -	0B FFFF
IO_0	0C 0000 -	0C 00FF
IO_1	0D 0000 -	0D 00FF
IO_2	0E 0000 -	0E 00FF
IO_3	0F 0000 -	0F 00FF

13. Skriv en subrutin (assembler) som kopierar ett sammanhängande block av data från en plats i minnet till en annan, icke överlappande, plats i minnet (`void copy_data(int *from, int *to, int no_data`). Funktionsparametrar skickas via stacken, `from` pekar på första elementet i blocket som ska kopieras, `to` pekar på första positionen i blocket dit data ska kopieras, och `no_data` indikerar hur många positioner som ska kopieras. Data i blocken samt `no_data` är 16-bitar. (Inga register ska påverkas av subrutinen) **12p**


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
GND
OUTPUTS