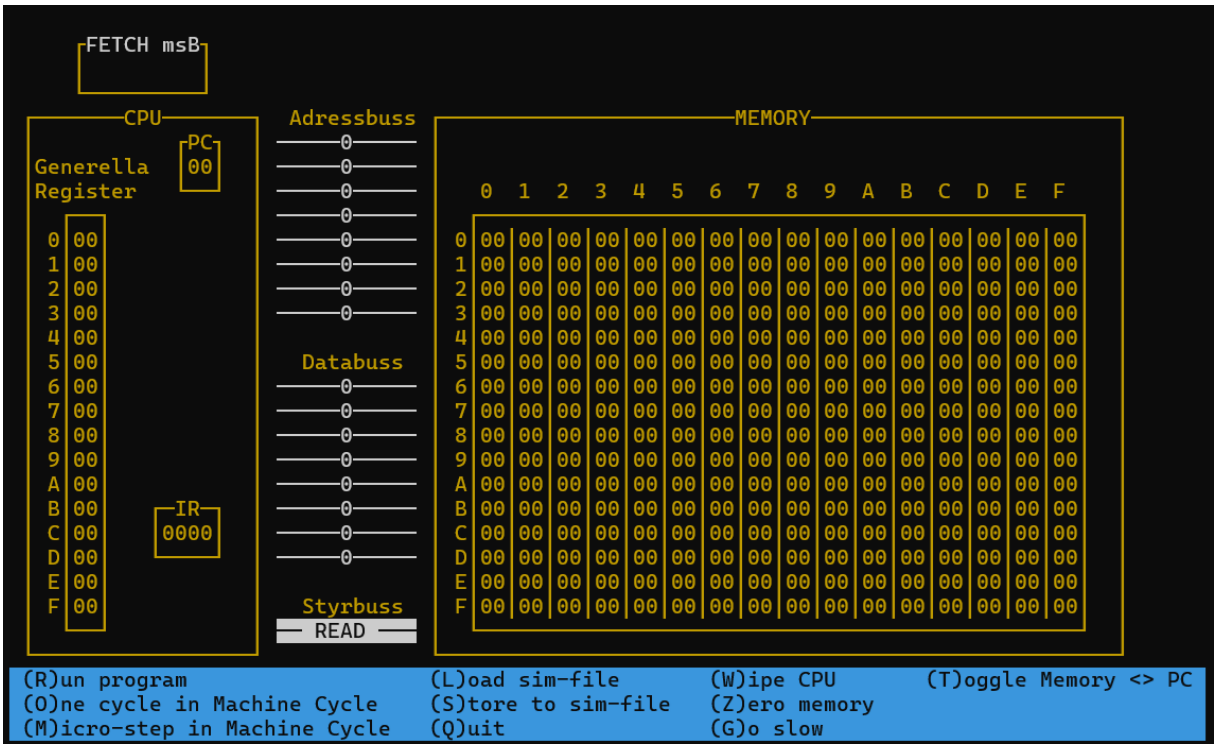
Laboration 2

Gunnar Landström
| HT24 DT155G_DT027G – Grundläggande datavetenskap |



Jan-Erik Jonssons maskinkodssimulator

Utrustning

Laboration utförd med Jan-Erik Jonsson maskinspråkssimulator med hjälp dess manual

Uppgifter

2.1

2.1 A) $8+5 = 13_{10}$ ($08_{16} + 05_{16} = 0D_{16}$)

- Program är placerat i MC: 00-09
- Indata är placerat i MC: F0 och F1
- Resultat är lagrat i MC: FF

10F0	Ladda R0 med MC F0 av tal 8 hexadecimalt ($\mathbf{08_{16}}$)
11F1	Ladda R1 med MC F1 av tal 5 hexadecimalt ($\mathbf{05_{16}}$)
5201	Addera R0 och R1 och placera resultatet i R2
32FF	Lagra resultatet i R2 på MC FF i hexadecimalt ($\mathbf{0D_{16}}$)
C000	Avsluta program

Maskinkod 2.1a)

:10:F0:11:F1:52:01:32:FF:C0:00:00:00:00:00:00:00

:08:05:00:00:00:00:00:00:00:00:00:00:00:00:00:0D

Alla tomma rader i maskinkoden är borttagna

Resultat 2.1a)

Programmet utför ett önskat resultat av att addera 8+5 och lagrar resultatet av additionen i begärd minnescell

2.1 B) 24 - 12 = 12_{10} (18_{16} + $F4_{16}$ = $0C_{16}$)

- Programmet är placerat i MC: 00-11
- Indata är placerat i MC: F0, F1, F2 och F3
- Resultat är lagrat i MC: FF

10F0	Ladda Register 0 med MC F0 av tal 24 hexadecimalt (**18_{16}**)
11F1	Ladda Register 1 med MC F1 av tal 12 hexadecimalt (**$0C_{16}$**)
1FF2	Ladda Register E med MC F2 av tal 1 hexadecimalt (**01_{16}**)
1EF3	Ladda register F med MC F3 av tal 255 hexadecimalt (**FF_{16}**)
921F	Ladda Register 2 med XOR ifrån Register 1 och Register F
532E	Addera Register 2 och Register E och placera resultatet i Register 3
5430	Addera Register 3 och Register 0 och placera resultatet i Register 4
34FF	Lagra resultatet i Register 2 på minnescell FF i hexadecimalt (**OC_{16}**)
C000	Avsluta Program

Maskinkod 2.1b)

```
:10:F0:11:F1:1E:F2:1F:F3:92:1F:53:2E:54:30:34:FF
:C0:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00
:18:0C:01:FF:00:00:00:00:00:00:00:00:00:00:00:0C
```

Alla tomma rader i maskinkoden är borttagna

Resultat 2.1b)

Programmet utför ett önskat resultat av att addera 24 med talet 12 konverterat till tvåkomplement för -12 och lagrar resultatet i minnescellen FF

2.1 C) -125 - 30 = -155_{10} ($7D_{16}$xor + $1E_{16}$xor = 65_{16})

- Program är placerat i MC: 00-15
- Indata är placerat i MC F0, F1, F2 och F3
- Resultat är lagrat i MC: FF

10F0	Ladda Register 0 med MC F0 för talet 125 (**$7D_{16}$**)
11F1	Ladda Register 1 med MC F1 för talet 30 (**$1E_{16}$**)
1EF2	Ladda Register E med MC F2 för talet 1 hexadecimalt (**01_{16}**)
1FF3	Ladda Register F med MC F3 för talet 255 hexadecimalt (**FF_{16}**)
920F	Ladda Register 2 med XOR ifrån Register 0 och Register F
532E	Addera Register 2 och Register E och placera resultatet i Register 3
941F	Ladda Register 4 med XOR ifrån Register 1 och Register F
554E	Addera Register 4 och Register E och placera resultatet i Register 5
5635	Addera Register 3 och Register 5 och placera resultatet i Register 6
36FF	Lagra resultatet i Register 6 på minnescell FF (**65_{16}**)
C000	Avsluta program

Maskinkod 2.1c)

:10:F0:11:F1:1E:F2:1F:F3:92:0F:53:2E:94:1F:55:4E

:56:35:36:FF:C0:00:00:00:00:00:00:00:00:00:00:00

:7D:1E:01:FF:00:00:00:00:00:00:00:00:00:00:00:65

Alla tomma rader i maskinkoden är borttagna

Resultat 2.1c)

Programmet utför ett oönskat resultat på grund av resursbrist då vi önskar ett resultat som kräver minst 9 bitar, men med en restriktion på 8 bitar så vi får en overflow där vi slår ihop två relativt stora negativa tal och får ett felaktigt stort positivt tal i retur.

2.2

2.2 a) $4*19 = 76_{10}$ ($13_{16} * 4 = 4C_{16}$)

- Programmet är skrivet i MC: 00-07
- Indata är placerat i MC: F0
- Resultat är lagrat i MC: FF

10F0	Ladda R0 med MC F0 med 19 hexadecimalt (**13_{16}**)
D002	Skifta R0 två gånger åt vänster
30FF	Lagra resultatet i R0 på MC FF (**$4C_{16}$**)
C000	Avsluta program

Maskinkod 2.2a)

:10:F0:D0:02:30:FF:C0:00:00:00:00:00:00:00:00:00

:13:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:4C

Alla tomma rader i maskinkoden är borttagna

Resultat 2.2a)

Programmet utför ett önskat resultat av att multiplicera 19 fyra gånger och lagrar resultatet 76 i minnescellen hexadecimalt.

2.2 b) $24_{10}/4 = (18_{16} / 4 = 06_{16})$

- Programmet är skrivet i MC: 00-07
- Indata är placerat i MC: F0
- Resultat är lagrat i MC: FF

10F0	Ladda R0 med MC xx med 24 hexadecimalt (**18_{16}**)
E002	Skifta R0 två gånger åt höger
30FF	Lagra resultatet i R0 på MC FF hexadecimalt (**06_{16}**)
C000	Avsluta program

Maskinkod 2.2b)

:10:F0:E0:02:30:FF:C0:00:00:00:00:00:00:00:00:00

:18:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:06

Alla tomma rader i maskinkoden är borttagna

Resultat 2.2b)

Programmet ger ett önskat resultat att dividera 24 med 4 och lagrar resultatet 6_{10} i minnescellen.

2.2 c) 19/2 = 9_{10} (13_{16} / 2 = 09_{16})

- Programmet är skrivet i MC: 00-07
- Indata är placerat i MC: 0F
- Resultat är lagrat i MC: FF

10F0	Ladda R0 med MC F0 med 19 hexadecimalt **(13_{16})**
E001	Skifta R0 en gång åt höger
30FF	Lagra resultatet i R0 på MC FF hexadecimalt (**09_{16}**)
C000	Avsluta program

Maskinkod 2.2c)

:10:F0:E0:01:30:FF:C0:00:00:00:00:00:00:00:00:00

:13:00:00:00:00:00:00:00:00:00:00:00:00:00:00:00:09

Alla tomma rader i maskinkoden är borttagna

Resultat 2.2c)

Vi får en overflow när en etta faller över kanten på höger sida så svaret blir 9 då vi ej använder oss utav float värden. Utan bara representerar ett heltal med 8 bitar.

2.3

a) 1010 1100 AND 0000 1111

- Programmet är placerat i MC: 00-09
- Indata är placerat i MC: F0 och F1
- Resultat lagras i MC: FF

1010 1100 (AC_{16})
<u>0000 1111</u> ($0F_{16}$)
0000 1100 ($0C_{16}$)

10F0	Ladda R0 med MC F0 för hexadecimalt **(AC_{16})**
11F1	Ladda R1 med MC F1 för hexadecimalt **($0F_{16}$)**
8201	Utför AND på bitmönstren i R0 och R1 och placera resultatet i R2
32FF	Lagra resultatet i R2 på MC FF **($0C_{16}$)**
C000	Avsluta program

Maskinkod 2.3a)

:10:F0:11:F1:82:01:32:FF:C0:00:00:00:00:00:00:00

:AC:0F:00:00:00:00:00:00:00:00:00:00:00:00:00:0C

Alla tomma rader i maskinkoden är borttagna

Resultat 2.3a)

Programmet utför ett önskat resultat av att ta två binära tal och utför en AND operation och lagrar resultatet i minnescellen.

2.3 b) 1010 1100 OR 0000 1111

- Programmet är placerat i MC: 00-09
- Indata är placerat i MC: F0 och F1
- Resultatet lagras i MC: FF

1010 1100 (AC_{16})
<u>0000 1111</u> ($0F_{16}$)
1010 1111 (AF_{16})

10F0	Ladda R0 med MC F0 för hexadecimalt **(AC_{16})**
11F1	Ladda R1 med MC F1 för hexadecimalt **($0F_{16}$)**
7201	Utför OR på bitmönstren i R0 och R1 och placera resultatet i R2
32FF	Lagra resultatet i R2 på MC FF **(AF_{16})**
C000	Avsluta program

Maskinkod 2.3b)

:10:F0:11:F1:72:01:32:FF:C0:00:00:00:00:00:00:00

:AC:0F:00:00:00:00:00:00:00:00:00:00:00:00:00:AF

Alla tomma rader i maskinkoden är borttagna

Resultat 2.3b)

Programmet utför ett önskat resultat av att ta två binära tal och utför en OR operation och lagrar resultatet i minnescellen.

2.3 c) 1010 1100 XOR 0000 1111

- Programmet är placerat i MC: 00-09
- Indata är placerat i MC: F0 och F1
- Resultatet lagras i MC: FF

1010 1100 (AC_{16})
<u>0000 1111</u> ($0F_{16}$)
1010 0011 ($A3_{16}$)

10F0	Ladda R0 med MC F0 för hexadecimalt **(AC_{16})**
11F1	Ladda R1 med MC F1 för hexadecimalt **($0F_{16}$)**
7201	Utför XOR på bitmönstren i R0 och R1 och placera resultatet i R2
32FF	Lagra resultatet i R2 på MC FF **(AF_{16})**
C000	Avsluta program

Maskinkod 2.3c)

:10:F0:11:F1:92:01:32:FF:C0:00:00:00:00:00:00:00

:AC:0F:00:00:00:00:00:00:00:00:00:00:00:00:00:A3

Alla tomma rader i maskinkoden är borttagna

Resultat 2.3c)

Programmet utför ett önskat resultat av att ta två binära tal och utför en XOR operation och lagrar resultatet i minnescellen.

3.2

Det största samt minsta positiva decimaltal vi kan representera i ett 8-bitars flyttal kodat enligt seeemmmm är:

- Max: $3{,}9375_{10}$ vilket skrivs $0111\ 1111_2$ som flyttal.
- Min: 0.00625_{10} vilket skrivs $0100\ 0001_2$ som flyttal.

Slutsatser

Vi kan i denna uppgift tydligt se att när det kommer till att utföra beräkningar med begränsade antal bitar så kan det fort bli markanta skillnader ifrån ett önskat resultat och det faktiska resultatet som maskinen matar ut. Den gör ju tekniskt sett inte fel, den får bara inte rätt förutsättningar för att lyckas med sin uppgift.