

Assembler för INTEL/AMD 64 bitar (x64) del 2

in real life

Mål

- Vanliga programkonstruktioner i assembler, t.ex
 - Jämförelse och villkorliga hopp
 - Iteration (while/for)
 - Selektion (if-else)
- Debugging
 - GDB
- Fler kodexempel med mix av assembler och högnivå
 - Assembler + C
 - C + assembler



Register

64-bitsregister	32-bitsregister	16-bitsregister	8-bitsregister
r <u>ax</u>	e <u>ax</u>	<u>ax</u>	al (ah, hög byte i ax)
r <u>bx</u>	e <u>bx</u>	<u>bx</u>	bl (bh, hög byte i bx)
r <u>cx</u>	e <u>cx</u>	<u>cx</u>	cl (ch, hög byte i cx)
r <u>dx</u>	e <u>dx</u>	<u>dx</u>	dl (dh, hög byte i dx)
r <u>si</u>	e <u>si</u>	<u>si</u>	<u>si</u> l
r <u>di</u>	e <u>di</u>	<u>di</u>	<u>di</u> l
r <u>bp</u>	e <u>bp</u>	<u>bp</u>	<u>bp</u> l
r <u>sp</u>	e <u>sp</u>	<u>sp</u>	spl
r8	r8d	r8w	r8b
r9	r9d	r9w	r9b
r10	r10d	r10w	r10 b
r11	r11d	r11w	r11b
r12	r12d	r12w	Addq = 64-bit add
r13	r13d	r13w	Addl = 32-bit add r_{13b}
r14	r14d	r14w	Addw = 16-bit add $Addb = 8$ -bit add
r15	r15d	r15w	r15b

Samtliga namn (utom specialfallen i parenteser) avser de lägsta bitarna



Anropskonventioner vid subrutin

Gäller AT&T (inte Windows-miljö)

```
OUT: rax
                                                          PUSH/POP: rbx
                                                               IN4: rcx
Heltalsparametrar skickas in (med call) i
                                                               IN3: rdx
%rdi, %rsi, %rdx, %rcx, %r8, %r9
                                                               IN2: rsi

    första parametern i %rdi

                                                               IN1: rdi
 andra parametern i %rsi
                                                          PUSH/POP: rbp
                                                     (Stackpekare): rsp

    Vid fler än 6 parametrar skickas resten via stacken

                                                               IN5: r8
                                                               IN6: r9
Flyttalsparametrar skickas in via %xmm (se komplett
                                                                    r10
(%rax innehåller antal %xmm som används)
                                                                    r11
                                                          PUSH/POP: r12
Returvärdet skickas i % rax som heltal (eller pekare)
                                                          PUSH/POP: r13
Vid flyttal: %xmm0
                                                          PUSH/POP: r14
                                                                    r15
```

 %rbp, %rbx och %r12 – %r14 måste sparas undan och återställas av en subrutin om deras värden förändras i rutinen



Programmering med subrutiner

Special-instruktioner:

Instruktion	Beskrivning	
call function	Subrutinanrop till <i>function (label),</i> återhoppsadress	
	läggs automatiskt på stacken	
ret	Retur från subrutin, återhoppsadress hämtas	
	automatiskt från stacken till pc	

Stack alignment

- Vid anrop av funktion måste stacken vara 16 bytes aligned
- Det innebär att stackpekarens värde måste vara en adress jämnt delbar med 16 (annars blir det segmentation fault)
- Speciellt viktigt vid anrop till externa funktioner/bibliotek, t.ex. printf() från C standardbibliotek (stdio.h)
- Ett funktionsanrop lägger återhoppsadressen på stacken (8 bytes) => stacken blir direkt unaligned
 - Inled med pushq \$0 för att få stacken 16 bytes aligned igen
 - eller minska stackpekare direkt med subq \$8, %rsp
 - Avsluta med popq till något register, eller med subq \$-8, %rsp (eller addq \$8, %rsp)



Vanliga programkonstruktioner i assembler

Suffix och datatyper

Utskrifter med printf (C standarbibliotek)

- Jämförelse och villkorliga hopp
- Iteration (while/for)

Selektion (if-else)



b, w, I och q suffix

```
C-kod:
            /* 32 bit */
int i=0;
long long j=1; /* 64 bit */
char c='A'; /* 8 bit */
i++;
j++;
c++;
INTEL-kod:
mov1 $0, %eax
             # 32 bit
movq $1, %rdi # 64 bit
movb $41, %cl
                #
                   8 bit
incl %eax
incq %rdi
incb %cl
```

```
32-bit:
             16-bits
                    8-bits
                                    OUT: rax
                    al
                              PUSH/POP: rbx
rax
      eax
             ax
rbx
      ebx
                    bl
             bx
                                     IN4: rcx
                    c1
rcx
      ecx
                                     IN3: rdx
rdx
      edx
             dx
                    d1
                                     IN2: rsi
      esi
                    sil
rsi
                                     IN1: rdi
rdi
      edi
                    dil
                              PUSH/POP: rbp
                    <u>bpl</u>
rbp
      ebp
             bp
                         (Stackpekare): rsp
      esp
                    spl
rsp
             sp
                                     IN5: r8
                                     IN6: r9
                                           r10
                                           r11
                              PUSH/POP: r12
                              PUSH/POP: r13
Addq = 64-bit addition
Add1 = 32-bit addition
                              PUSH/POP: r14
Addw = 16-bit addition
                                           r15
Addb = 8-bit addition
```

OBS! "Integer" kan variera med applikation, ofta 32 bitar men ibland 16 bitar.



printf()

32-bit:

16-bits 8-bits

```
OUT: rax
                                                          al
                                                                 PUSH/POP: rbx
                                          rax
                                               eax
                                          rbx
                                               ebx
                                                    <u>bx</u>
                                                         bl
C-kod:
                                                                      IN4: rcx
                                                          c1
                                          <u>rcx</u>
                                               ecx
int i=10;
                         /* 32 bit */
                                                                      IN3: rdx
                                               edx
                                                          d1
                                          <u>rdx</u>
                                                    dx
printf ("%d \n", i);
                                                                      IN2: rsi
                                          rsi
                                               esi
                                                          sil
                                                                      IN1: rdi
                                               edi
                                          rdi
                                                          dil
INTEL-kod:
                                                                 PUSH/POP: rbp
                                               ebp
                                                    <u>bp</u>
                                                          <u>bpl</u>
                                          rbp
                                                             (Stackpekare): rsp
.data
                                                          spl
                                          rsp
                                               esp
                                                                      IN5: r8
string: .asciz "%d \n"
                                                                      IN6: r9
.text
                                                                           r10
movl $10, %ecx
                          # 32 bit
                                                                           r11
leaq string, %rdi
                          # %rdi pekar på string
                                                                  PUSH/POP:
                                                                           r12
                             Ladda inparameter 1 (ovan)
                                                                  PUSH/POP:
                                                                           r13
movl %ecx, %esi
                          # Ladda inparameter 2
                                                                 PUSH/POP: r14
(movq $0, %rax
                            # Nollställ utparameter)
                                                                           r15
call printf
                          # Tänk på stack alignment!
```



while loop

32-bit:

16-bits

8-bits

OUT: rax

```
PUSH/POP: rbx
                                                                al
                                                    eax
                                               rax
                                                          ax
                                                    ebx
                                                                bl
                                               rbx
                                                          bx
                                                                              IN4: rcx
C-kod:
                                                                c1
                                                    ecx
                                                          CX
                                               rcx
                                                                              IN3: rdx
while (i<10)
                                                                d1
                                                    edx
                                                          dx
                                               rdx
                                                                              IN2: rsi
                                               rsi
                                                    esi
                                                                sil
                                                                              IN1: rdi
//CODE//
                                               rdi
                                                    edi
                                                                dil
                                                                        PUSH/POP: rbp
                                               rbp
                                                    ebp
                                                                <u>bpl</u>
                                                          <u>bp</u>
                                                                   (Stackpekare): rsp
                                                                spl
                                               rsp
                                                    esp
                                                          sp
                                                                              IN5: r8
INTEL-kod:
                                                                              IN6: r9
                                                                                   r10
whileloop:
                                                                                   r11
    cmpl $10, %ecx
                              # Beräkna %ecx-10 (eflags)
                                                                        PUSH/POP:
                                                                                  r12
    jge endwhileloop
                                                                        PUSH/POP:
                                                                                  r13
    //CODE//
                                                                        PUSH/POP: r14
                                                                                   r15
    jmp whileloop
endwhileloop:
```



endforloop:

for loop

```
32-bit:
                                                             16-bits
                                                                   8-bits
                                                                                  OUT: rax
                                                                   al
                                                                            PUSH/POP: rbx
                                                 rax
                                                       eax
C-kod:
                                                 rbx
                                                       ebx
                                                             <u>bx</u>
                                                                   bl
                                                                                  IN4: rcx
for (int i=0;i<5;i++)</pre>
                                                 rcx
                                                       ecx
                                                             CX
                                                                   c1
                                                                                  IN3: rdx
                                                       <u>edx</u>
                                                                   d1
                                                 rdx
                                                             dx
                                                                                  IN2: rsi
                                                 rsi
                                                       esi
                                                             Si
                                                                   Sil
                                                                                  IN1: rdi
//CODE//
                                                 rdi
                                                       edi
                                                             di
                                                                   dil
                                                                            PUSH/POP: rbp
                                                 rbp
                                                       ebp
                                                                   <u>bpl</u>
                                                             <u>bp</u>
                                                                       (Stackpekare): rsp
                                                                   spl
                                                 rsp
                                                       esp
                                                             sp
                                                                                  IN5: r8
INTEL-kod:
                                                                                  IN6: r9
    mov1 $0, %ecx
                                                                                       r10
                                                                                       r11
forloop:
                                                                            PUSH/POP:
                                                                                       r12
                                  # Beräkna %ecx-5 (eflags)
    cmpl $5, %ecx
                                                                            PUSH/POP:
                                                                                       r13
    jge endforloop
                                                                            PUSH/POP: r14
//CODE//
                                                                                       r15
    incl %ecx
    jmp forloop
```



if..else sats

```
C-kod:
if (i==2)
//CODE1
else
//CODE2
INTEL-kod:
   cmpl $2, %ecx
   jne elsetrue
   //CODE1//
   jmp endif
elsetrue:
   //CODE2//
endif:
```

```
16-bits
                            32-bit:
                                           8-bits
                                                             OUT: rax
                                                      PUSH/POP: rbx
                                           al
                            eax
                                   <u>ax</u>
                     rax
                     rbx
                            ebx
                                   <u>bx</u>
                                           bl
                                                             IN4: rcx
                                           c1
                            ecx
                                   CX
                                                             IN3: rdx
                     rcx
                     rdx
                            edx
                                   dx
                                           d1
                                                             IN2: rsi
                                   si
                     rsi
                            esi
                                           sil
                                                             IN1: rdi
                     rdi
                            edi
                                   di
                                           dil
                                                      PUSH/POP: rbp
                            ebp
                                           <u>bpl</u>
                     rbp
                                   <u>bp</u>
                                                (Stackpekare): rsp
                                           spl
                     rsp
                            esp
                                   sp
                                                             IN5: r8
                                                             IN6: r9
                                                                    r10
                                                                    r11
                                                      PUSH/POP:
                                                                   r12
# Beräkna %ecx-2 (eflags)
                                                      PUSH/POP: r13
                                                      PUSH/POP: r14
                                                                    r15
```



ret

Funktion greater()

16-bits

32-bit:

8-bits

OUT: rax

```
PUSH/POP: rbx
                                                                  al
                                                rax
                                                      eax
                                                            ax
                                                rbx
                                                      ebx
                                                            bx
                                                                  bl
C-kod:
                                                                                IN4: rcx
char greater (int *a, int *b)
                                                                  c1
                                                      ecx
                                                            CX
                                                rcx
                                                                                IN3: rdx
                                                                  d1
                                                rdx
                                                      edx
                                                            dx
                                                                                IN2: rsi
if (*a>*b)
                                                                  sil
                                                rsi
                                                      esi
                                                                                IN1: rdi
  return 1;
                                                rdi
                                                      edi
                                                                  dil
                                                                          PUSH/POP: rbp
else
                                                      ebp
                                                                  bpl
                                                rbp
                                                            bp
                                                                      (Stackpekare): rsp
  return 0;
                                                                  spl
                                                rsp
                                                      esp
                                                            sp
                                                                                IN5: r8
                                                                                IN6: r9
INTEL-kod:
                                                                                      r10
greater:
                                                                                     r11
   movl (%rdi), %ecx # Läs in inparameter 1 från minne till %ecx
                                                                          PUSH/POP:
   movl (%rsi), %edx # Läs in inparameter 2 från minne till %edx
                                                                                     r12
   cmpl %ecx, %edx
                        # Beräkna %edx-%ecx och uppdatera eflags
                                                                          PUSH/POP: r13
   jg gtr
                                                                          PUSH/POP: r14
   movb $0, %al
                        # %al = lägsta 8-bitar av %rax
                                                                                     r15
   ret
qtr:
   movb $1, %al
                        # %al = lägsta 8-bitar av %rax
```



Debugging med GDB

Se sida i Canvas-modulen "Laborationer":

GDB debugtips



Debugging med GDB

Utdrag ur GDB debugtips:

GDB är en debugger som finns att installera på er Linux-maskin. Det är en textbaserad debugger, men har alla de funktioner man känner igen från andra utvecklingsmiljöer, t.ex. VS Code. Det man saknar är förstås ett grafiskt användargränssnitt, men efter ett tags användning fungerar detta utmärkt.

För generell info om GDB se till exempel: https://sourceware.org/qdb/current/onlinedocs/qdb.html

Här kommer lite extra tips för hur man kan debugga sitt program (t.ex. assembler- och/eller C-program) med GDB.

Speciellt kommer vi att använda något som heter **GDB TUI** (**T**ext **U**ser **I**nterface), se mer på
https://sourceware.org/gdb/current/onlinedocs/gdb.html/TUI.html



Debugging med GDB - exempel

```
TERMINAL

    gdb - exercise4 + ∨ □ 
    □ ··· ∨

 -Register group: general-
                0x4011af
                                     4198831
                                                                  rbx
                                                                                 0x0
                                                                                                      0
                                                                                 0x7fffffffdc88
 rcx
                0x403e18
                                     4210200
                                                                  rdx
                                                                                                      140737488346248
 rsi
                0x7fffffffdc78
                                                                  rdi
                                    140737488346232
                                                                                 0x1
                0x7fffffffdb60
                                    0x7fffffffdb60
                                                                                 0x7fffffffdb40
                                                                                                      0x7fffffffdb40
 rbp
                                                                  rsp
 r8
                0x7ffff7fa6f10
                                    140737353772816
                                                                  r9
                                                                                 0x7ffff7fc9040
                                                                                                      140737353912384
 r10
                0x7ffff7fc3908
                                     140737353890056
                                                                  r11
                                                                                  0x7ffff7fde660
                                                                                                      140737353999968
 r12
                0x7fffffffdc78
                                     140737488346232
                                                                  r13
                                                                                  0x4011af
                                                                                                      4198831
 r14
                0x403e18
                                    4210200
                                                                  r15
                                                                                 0x7ffff7ffd040
                                                                                                      140737354125376
 rip
                0x4011bb
                                    0x4011bb <main+12>
                                                                  eflags
                                                                                  0x202
                                                                                                      [ IF ]
 cs
                                                                                  0x2b
                0x33
 -main4.c-
         1 #include <stdio.h>
         2 int readInt(char *);
         3 int main()
                char str [12];
                int res;
                printf("Mata in ett tal! Avsluta med #!\n");
                fgets(str, 12, stdin);
                res = readInt(str);
        10
                printf("Talet r: %d \n", res);
                return 0;
multi-thre Thread 0x7ffff7d887 In: main
                                                                                                                         PC: 0x4011bb
(gdb) layout regs
(gdb) start
Temporary breakpoint 1 at 0x4011bb: file main4.c, line 4.
Starting program: /home/pin/proj/datorteknik/lectures/intel/exercise4/prog
[Thread debugging using libthread db enabled]
Using host libthread db library "/lib/x86 64-linux-gnu/libthread db.so.1".
Temporary breakpoint 1, main () at main4.c:4
(gdb)
```



Fler kodexempel

- Fyra uppgifter
- Uppgift 3 och 4 viktiga inför Laboration 3!

Lösningar i separat presentation och i kod.



OUT: rax

```
PUSH/POP: rbx
                                                                       IN4: rcx
Skriv en subrutin, max(x,y), som bestämmer vilket av
                                                                       IN3: rdx
två heltal som är störst där
                                                                       IN2: rsi
        x = första heltalet
                                                                       IN1: rdi
        y = andra heltalet
        returvärde = det största av de två talen x och y.
                                                                  PUSH/POP: rbp
Rutinen ska anropas från följande C-program:
                                                             (Stackpekare): rsp
                                                                       IN5: r8
                                                                       IN6: r9
#include <stdio.h>
                                         Noteringar:
int max(int, int);
                                                                            r10
                                         x och y = 32-bit integer
                                         X = edi
                                                                            r11
int main()
                                         V = %esi
                                                                  PUSH/POP:
                                                                            r12
                                         max = %eax
   int x,y,res;
                                                                  PUSH/POP: r13
   printf("Mata in två heltal\n")
                                                                  PUSH/POP: r14
   scanf("%d", &x);
   scanf("%d", &y);
                                                                            r15
   res = max(x,y);
   printf("Talet %d var störst av dem\n", res);
   return 0;
```



OUT: rax PUSH/POP: rbx IN4: rcx IN3: rdx IN2: rsi IN1: rdi PUSH/POP: rbp (Stackpekare): rsp IN5: r8 IN6: r9 r10 r11 PUSH/POP: r12 PUSH/POP: r13 PUSH/POP: r14 r15

```
Antag att vi har en sekvens av n st positiva tal (array). Skriv en subrutin,maxNum, som ger oss det största talet i sekvensen.
```

Argument 1: Adress till talsekvensen

Argument 2: Antalet tal i sekvensen

Returvärde: Det största av talen i sekvensen

Rutinen ska ingå i ett bibliotek som ska anropas från följande C-progr.:

```
#include <stdio.h>
int maxNum(int *, int);
int main()
{
   int vect[5];
   int res;

   printf("Mata in fem heltal\n");
   scanf("%d", &vect[0]);
   scanf("%d", &vect[1]);
   scanf("%d", &vect[2]);
   scanf("%d", &vect[3]);
   scanf("%d", &vect[4]);
   res = maxNum(vect,5);
   printf("Talet %d var störst av dem\n", res);
   return 0;
}
```

Noteringar:

vect[5] = 5 st 32-bit integer
res = 32-bit integer
(%rdi) pekar på vect[0]
%esi = 5
maxNum = %eax



Vi har en teckensträng som startar med siffror avslutas med "icke-siffra".

Ex: "1234<u>X</u>", "122<u>Y</u>", "13323_" eller "123123<u>C</u>"

Skriv en rutin, readInt, som går igenom strängen och <u>returnerar ett heltal</u> (det tal vars representation utgörs av siffrorna i strängen).

Argument: Adress till strängen

Returvärde: Utläst heltal

Rutinen ska kunna anropas från följande C-program:

```
#include <stdio.h>
int readInt(char *);

int main()
{
   char str [10];
   int res;
   printf("Mata in ett tal! Avsluta med #!\n");
   fgets(str, 12, stdin);
   res = readInt(str);
   printf("Talet är: %d \n", res);
   return 0;
```

Noteringar:

```
str[10] = 10 st char
res = 32-bit integer
(%rdi) pekar på str[0]
readInt = %eax
```



Vi har en teckensträng som startar med siffror avslutas med "icke-siffra".

Den kan innehålla ett godtyckligt antal blanktecken (noll eller flera) framför talet samt att talet kan inledas med ett tecken '+' eller '-' före första siffran.

Ex: " +1234X", " -122Y", " 13323 " eller "123123C"

Modifiera uppgift 3 med den röda inringningen!

Att-göra-lista:

- Införa en teckenindikator (ifall vi upptäcker minustecken)
- Leta upp blanktecken och kasta dem
- Leta upp ev. '+'
- Leta upp ev. '-'
- Om minus: Teckenindikator = 1, annars 0
- Kör uppgift 3
- · Ändra ev. tecken till negativt

Noteringar:

str[10] = 10 st char res = 32-bit integer

(%rdi) pekar på str[0]

readInt = %eax