Raspberry PI ASM 64 bit

Bartolomeo Montrucchio

Politecnico di Torino

Dipartimento di Automatica e Informatica (DAUIN)

Torino - Italy

Introduction

- See README in Materiale/Laboratorio_Raspberry
- https://blahcat.github.io/2018/01/07/building-a-debian-stretch-qemu-image-for-aarch64/
- Take image arm64_stretch from:
  - https://mega.nz/#F!oMoVzQaJ!iS73iiQQ3t_6HuE-XpnyaA
  - ./start.sh
  - Cortex A-53 is fully cross-emulated

ARM 64 bit

- Why porting to 64 bits?
  - https://community.arm.com/cfs-file/__key/telligent-evolution-components-attachments/01-2142-00-00-00-00-52-01/Porting-to-ARM-64_2D00_bit.pdf
- need to address more than 4GB of memory
- need for wider registers and greater accuracy of 64-bit data processing
- need for larger register set.

Basic example (gcc based) 1/3

```
int main()
{
        int i;
        i = i + 5;
        return 9;
}
```

```
$gcc –S –o first.s first.c

$gcc –o first first.o
$./first
```

ATTENTION: there are two different syntaxes for ASM 64 bit see this example

Basic example (gcc based) 2/3

```
root@debian-aarch64:~/asm/constructs_64bit# cat first.s
        .arch armv8-a
        .file   "first.c"
        .text
        .align  2
        .global main
        .type   main, %function
main:
        sub     sp, sp, #16
        ldr     w0, [sp, 12]
        add     w0, w0, 5
        str     w0, [sp, 12]
        mov     w0, 9
        add     sp, sp, 16
        ret
        .size   main, .-main
        .ident  "GCC: (Debian 6.3.0-18) 6.3.0 20170516"
        .section .note.GNU-stack,"",@progbits
```

Basic example (gcc based) rewritten 3/3

```
root@debian-aarch64:~/asm# cat test.s
// test.s
// compile with gcc test.s
.text
.globl main
main:
  add w0, w0, #5   // w0 ← w0 + 5
  mov w0, #9
  ret              // return from main

root@debian-aarch64:~/asm# gcc test.s
root@debian-aarch64:~/asm# ./a.out
root@debian-aarch64:~/asm# echo $?
9
```

- Different syntax!

If 1/2

```
# cat if.c
int main()
{
        int a;
        a = 3;
        if (a == 4)
        {
                a+= 8;
        }
        else
        {
                a+= 20;
        }
        return a;
}
```

```
main:
        sub     sp, sp, #16
        mov     w0, 3
        str     w0, [sp, 12]
        ldr     w0, [sp, 12]
        cmp     w0, 4
        bne     .L2
        ldr     w0, [sp, 12]
        add     w0, w0, 8
        str     w0, [sp, 12]
        b       .L3
.L2:
        ldr     w0, [sp, 12]
        add     w0, w0, 20
        str     w0, [sp, 12]
.L3:
        ldr     w0, [sp, 12]
        add     sp, sp, 16
        ret
```

If 2/2

$ gcc -O3 -S -o if_O3.s if.c
NOTE: same result also with –O1 and –O2!!

```
root@debian-aarch64:~/asm/constructs_64bit# cat if_O3.s
.........
main:
        mov     w0, 23
        ret
        .size   main, .-main
        .ident  "GCC: (Debian 6.3.0-18) 6.3.0 20170516"
        .section        .note.GNU-stack,"",@progbits
```

For

```
int main()
{
        int i;
        int j=0;

        for (i=0; i<10; i++)
        {
                j++;
        }

        return j;
}
```

```
main:
sub     sp, sp, #16
str     wzr, [sp, 8]
str     wzr, [sp, 12]
b       .L2
.L3:
ldr     w0, [sp, 8]
add     w0, w0, 1
str     w0, [sp, 8]
ldr     w0, [sp, 12]
add     w0, w0, 1
str     w0, [sp, 12]
.L2:
ldr     w0, [sp, 12]
cmp     w0, 9
ble     .L3
ldr     w0, [sp, 8]
add     sp, sp, 16
ret
```

While

```
int main()
{
        int j=10;

                int i=0;
                while (i<10)
                {
                        j+=3;

                        i++;
                }
}
```

```
main:
sub     sp, sp, #16
mov     w0, 10
str     w0, [sp, 12]
str     wzr, [sp, 8]
b       .L2
.L3:
ldr     w0, [sp, 12]
add     w0, w0, 3
str     w0, [sp, 12]
ldr     w0, [sp, 8]
add     w0, w0, 1
str     w0, [sp, 8]
.L2:
ldr     w0, [sp, 8]
cmp     w0, 9
ble     .L3
mov     w0, 0
add     sp, sp, 16
ret
```

Multiplication

```
int main()
{
        long a=7;
// long is 32 bits on ARM v7
// and 64 bits on ARM v8
        long long b=12;
// long long is always 64 bits

        a *= a;
        b *= b;
}
```

```
sub     sp, sp, #16
mov     x0, 7
str     x0, [sp, 8]
mov     x0, 12
str     x0, [sp]
ldr     x1, [sp, 8]
ldr     x0, [sp, 8]
mul     x0, x1, x0
str     x0, [sp, 8]
ldr     x1, [sp]
ldr     x0, [sp]
mul     x0, x1, x0
str     x0, [sp]
mov     w0, 0
add     sp, sp, 16
ret
```

Division

```
int main()
{
        long a=42;
        long b=7;
        long c=0;

    c=a/b;
        return c;
}
```

```
sub   sp, sp, #32
mov   x0, 42
str   x0, [sp, 24]
mov   x0, 7
str   x0, [sp, 16]
str   xzr, [sp, 8]
ldr   x1, [sp, 24]
ldr   x0, [sp, 16]
sdiv  x0, x1, x0
str   x0, [sp, 8]
ldr   x0, [sp, 8]
add   sp, sp, 32
ret
```