Prof. Dr. Armin Biere
Dr. Mathias Fleury

Freiburg, 18/11/2021

Computer Architecture
Exercise Sheet 3 (version 2022-7)

Exercise 1

a) Transform the following C code (where 0x is a prefix to indicate the numer is hexadecimal) into RISC-V assembly.

```
int a = 0xDEADBEEF;
```

b) Transform the following C code (where 0x is a prefix to indicate the numer is hexadecimal) into RISC-V assembly:

```
unsigned int a /*comes from context*/; signed int b;
if (a <= (unsigned)INT_MAX)
  b = (int) a;
else
  b = -1;
```

Compile the following code to RISC-V:

```
int a /*comes from context*/; signed int b;
if (a <= (int)INT_MAX)
  b = (int) a;
else
  b = -1;
```

How can it be optimized? In which of the two previous programs can the condition be replaced by `a < INT_MAX+1`?

c) Transform the following pseudocode into proper assembly

```
goto instruction such that PC = PC + 0xBADEAFFE;
goto instruction such that PC = PC + 2047;
```

d) For the exam we might instead ask to fill out the missing parts in the assembler program.

Original C program:

```
int a = 0xDEADB00F;
```

Corresponding machine code and assembler program:

```
0: deadb7b7           lui a5,0xdeadb
4: ___78793           addi a5,a5,___
```

Exercise 2

We consider the following C structure:

```
struct {
  int x;
  int y;
} POINT;
```

Assume that there are no padding bits, that `x` is before y, and that there is no extra alignment condition. Therefore, the memory representation is `|x|y|`.

a) What is the size of the structure in words?

b) We want to sort coordinates in an array

```
int compare (struct POINT * a, struct POINT * b)
{
  if (a->x < b->x)
      return -1;
  if (a->x == b->x) {
      if (a->y < b->y) return -1;
      else if (a->y == b->y) return 0;
      else return 1;
  }
  return 1;
}
```

We provide you an extract of the RISC-V code. Complete the holes wherever needed and annotate it to match the C code.

```
compare:                                # @compare
  addi    sp, sp, ____
  sw      ra, 28(sp)
  sw      s0, 24(sp)
  addi    s0, sp, 32
  sw      a0, -16(s0)
  sw      a1, -20(s0)
  lw      a0, -16(s0)

  mv      a2, a0
  lw      a3, 0(a0)
  lw      a4, 0(a1)
  li      a0, -1
  blt     a3, a4, ___
  li      a0, 1
  bne     a3, a4, ___
  lw      a2, 4(a2)
  lw      a1, 4(a1)
  li      a0, -1
  blt     a2, a1, ____
  xor     a0, a2, a1
  snez    a0, a0

end:
  lw      ra, 28(sp)
  lw      s0, 24(sp)
  addi    sp, sp, ____
  ret
```

c) We want to compare an array of coordinates (`POINT*`) to having two array of coordinates:

```
struct {
  int* xs;
  int* ys;
} COORDINATES;
```

We have the following compare function:

```
int compare (COORDINATES a, int i, int j)
{

  if (a.xs[i] < a.xs[j])
     return -1;
  if (a.xs[i] == a.xs[j]) {
     if (a.ys[i] < a.ys[j]) return -1;
     else if (a.ys[i] == a.ys[j]) return 0;
   }
   return 1;
}
```

Annotate and complete it:

```
compare:                                # @compare
        slli    a2, a2, ____
        add     a4, a0, a2
        lw      a4, 0(a4)
        slli    a3, a3, ____
        add     a0, a0, a3
        lw      a5, 0(a0)
        li      a0, -1
        blt     a4, a5, ______
        bne     a4, a5, ______
        add     a2, a2, a1
        lw      a2, 0(a2)
        add     a1, a1, a3
        lw      a1, 0(a1)
        blt     a2, a1, _____
        bne     a2, a1, _____
        li      a0, 0
        ret
.LBB0_5:
        li      a0, 1
.LBB0_6:
        ret
```

d) Why is there no operation on the stack pointer `sp`?

(`http://www.catb.org/esr/structure-packing/` is *the* reference for everything about padding and alignment).

Exercise 3

You are working for a bank. This bank stores the amount of money on your account *two* numbers, the integer amount in euros and the interger amount of cents.

Assume that the memory location of the two amounts is in `a0` (euros) and `a1` (cent).

a) Write the function that checks that you have enough money on your account to pay the amount X specified in `a2` (euros) and `a3` (cents). To simplify the code, we assume that (i) `a2` is positive and (ii) `a3` is zero and (iii) there is no overflow/undeflow.

b) Write the function to remove the amount of money under the same assumptions.

c) Yours account has 100 euros and two stores are trying to debit money at the same time: 50 euro and 51,50 euros. They first check whther there is enough moeny, then debit it. You forget to make the operations atomic. List all the possible amount of money on your account, assuming that writing is atomic (only one write to main memory happens at a time).

100 - 50 = 50
100 - 51.50 = 48.5

No submission is needed. The exercise sheet will be discussed on November 24th, 2022 in class and in the online exercise session, 10:00 at:
https://uni-freiburg.zoom.us/j/65775356475?pwd=dmUvei8ybDN4RFlmT1JUZnRtYlBGZz09