Computer Engineering Course 2023/2024

Report 1

Date: 2023/10/18 **Name 1:** Rahaf Alabrsh **Name 2:** Christian Panov **Name 3:** Cora Zeitler

Accessing the Pis

Output hostname:
`rspi06.inf-ra.uni-jena.de`

Output lscpu:

```
Architecture:            aarch64
  CPU op-mode(s):        32-bit, 64-bit
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               ARM
  Model name:            Cortex-A72
    Model:               3
    Thread(s) per core:  1
    Core(s) per cluster: 4
    Socket(s):           -
    Cluster(s):          1
    Stepping:            r0p3
    CPU max MHz:         1500,0000
    CPU min MHz:         600,0000
    BogoMIPS:            108.00
    Flags:               fp asimd evtstrm crc32 cpuid
Caches (sum of all):
  L1d:                   128 KiB (4 instances)
  L1i:                   192 KiB (4 instances)
  L2:                    1 MiB (1 instance)
Vulnerabilities:
  Itlb multihit:         Not affected
  L1tf:                  Not affected
  Mds:                   Not affected
  Meltdown:              Not affected
  Mmio stale data:       Not affected
  Retbleed:              Not affected
  Spec store bypass:     Vulnerable
  Spectre v1:            Mitigation; __user pointer sanitization
  Spectre v2:            Vulnerable
  Srbds:                 Not affected
  Tsx async abort:       Not affected
```

Working with Bits

name	type	value	bit value	explanation
`l_data1`	`unsigned char`	`1`	`00000001`	We selected 8 bits as that is the size of the char datatype. The leading zeroes represent the empty bits.
`l_data2`	`unsigned char`	`255`	`11111111`	This is simply 255 in binary: the largest number that can be stored in eight bits.
`l_data3`	`unsigned char`	`l_data2 + 1`	`00000000`	256 is beyond the boundary of numbers that can be stored as char datatype. Adding 1 to the char maximum creates an overflow, flipping all the bits back to 0.
`l_data4`	`unsigned char`	`0xF0`	`11110000`	$F_{16} = 15_{10} = 1111_2$. A single hex digit can be converted into a group of four bits. Zero is "0000". So it's just those digits in binary in the same order.
`l_data5`	`unsigned char`	`0b10111`	`00010111`	This is already in binary, so the bit output is the same (with additional leading zeroes because of the eight bits).

l_data6	unsigned char	'J'	01001010	Characters are saved as actual numbers (their ASCII code). For J that's 112_{10} = $4A_{16}$ = 1001010_2 (the most significant bit being represented as leading 0).
l_data7	char	-11	11110101	For signed datatypes, the most significant bit is reserved for the sign: 1 for negative, 0 for positive. Positive 11_{10} = 00001011_2. Bitwise NOT leads to 11110100, the *one's complement*. Adding 1 to the one's complement yields the *two's complement*; the usual representation of negative numbers in binary. Thus -11_{10} is 11110101_2 when using eight bits.
l_data8	unsigned int	1u << 7	00000000000000000000000010000000	1u is an unsigned 1. "a << b" shifts the bits of number a by b places to the left (inserting zeroes on the right). The 1 is now on the eigth position.
l_data9	unsigned int	l_data8 << 23	01000000000000000000000000000000	The previous number is shifted left another 23 times.
l_data10	unsigned int	0xFFFFFFFF >> 3	00011111111111111111111111111111	Each hex digit corresponds to a 4-group of bits. F_{16} = 1111_2. Thus $FFFFFFFF_{16}$ = $11111111111111111111111111111111_2$. Shifted right three times, it inserts three zeroes to the left. Because the bitshift operator is acyclical, bits that shift "out of bounds" are lost.
l_data11	unsigned int	0b1001 ^ 0b11011	00000000000000000000000000010010	^ is a bitwise XOR. 01001 XOR 11011 yields 10010.
l_data12	unsigned int	~0b0011	11111111111111111111111111111100	NOT is true except for the two lowest bits.
l_data13	unsigned int	0xB0 & 0b1010101	00000000000000000000000000010000	Bitwise AND. $B0_{16}$ = 10110000_2. 10110000 AND 01010101 = 00010000 (only the fifth bit is 1 in both numbers).
l_data14	unsigned int	0b011 | 0b110	00000000000000000000000000000111	011 OR 110 yields 111.
l_data15	unsigned int	0x7c0	00000000000000000000011111000000	$7C0_{16}$ = 011111000000_2.
l_data16	int	-1984	11111111111111111111100001000000	1984_{10} = 11111000000_2. Its 32-bit one's complement is 11111111111111111111100000111111. Adding 1 to that gives the two's complement 11111111111111111111100001000000.