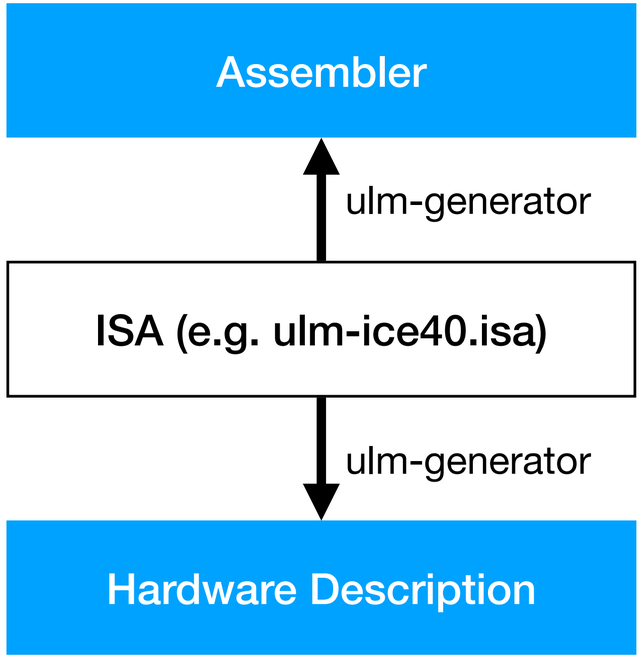
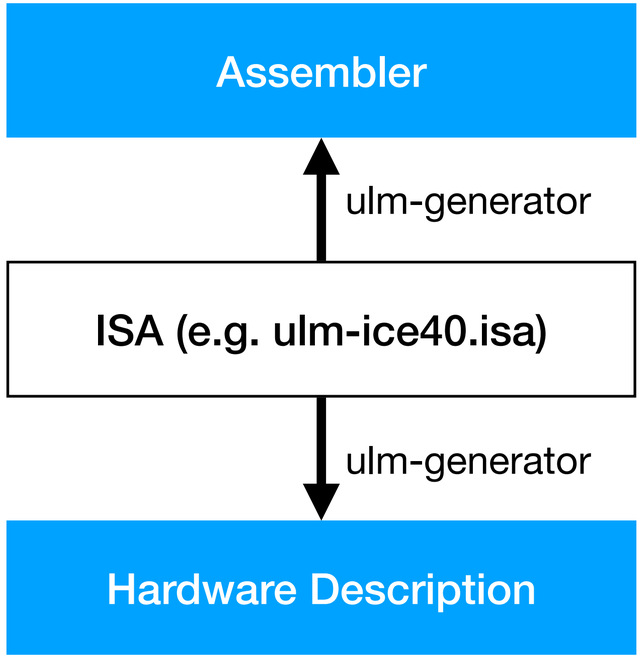
ULM: Ulm Lecture Machine (Bottom-Up Approach)

Assembly Programming: First Steps

ISA (Instruction Set Architecture)
Interface between Software and Hardware



ISA (Instruction Set Architecture)
Interface between Software and Hardware

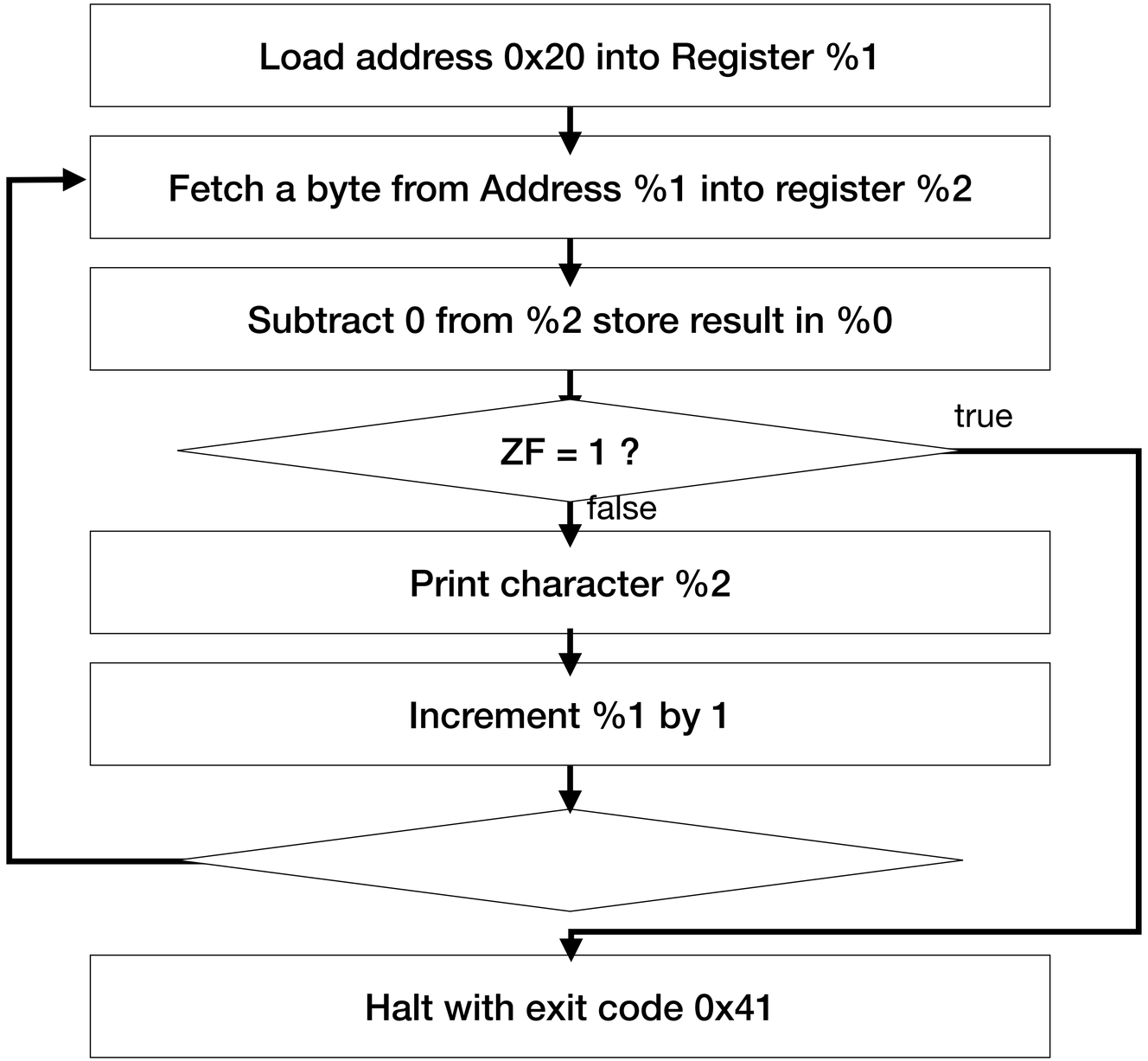



```
U16_R_R     (OP u 8) (z u 4) (y u 4) (imm u 16)
R_R_R       (OP u 8) (z u 4) (y u 4) (x u 4)

# ...

0x12 U16_R_R
#   Adds the zero extended immediate value \\texttt{imm} to register
#   \\texttt{%y} and stores the result in register \\texttt{%z}.
:   addq imm, %y, %z
    ulm_add64(imm, ulm_regVal(y), z);

0x13 R_R_R
#   Adds register \\texttt{%x} to register and stores the result in
#   register \\texttt{%z}.
:   addq %x, %y, %z
:   movq %x, %z
    ulm_add64(ulm_regVal(x), ulm_regVal(y), z);
```


Load address 0x20 into Register %1
Fetch a byte from Address %1 into register %2
Subtract 0 from %2 store result in %0
ZF = 1 ?
true
false
Print character %2
Increment %1 by 1
Halt with exit code 0x41

```
    loadz   0x20,   %1
    movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      16
    putc    %2
    addq    1,      %1,     %1
    jmp     -20
    halt    0x41
```

```
    loadz   0x20,   %1
    movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      16
    putc    %2
    addq    1,      %1,     %1
    jmp     -20
    halt    0x41
```

```
#TEXT 4
0x0000000000000000: 10 10 00 20 #     loadz 32, %1
0x0000000000000004: 20 21 00 00 #     movzbq (%1), %2
0x0000000000000008: 14 02 00 00 #     subq 0, %2, %0
0x000000000000000C: 04 00 00 04 #     jz 16
0x0000000000000010: 30 20 00 00 #     putc %2
0x0000000000000014: 12 11 00 01 #     addq 1, %1, %1
0x0000000000000018: 05 FF FF FB #     jmp -20
0x000000000000001C: 01 41 00 00 #     halt 65
#DATA 1
#BSS 1 0
#SYMTAB
```

```
    loadz   0x20,   %1
    movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      16
    putc    %2
    addq    1,      %1,     %1
    jmp     -20
    halt    0x41
    .string "hello, world!\n"
```

```
    loadz   0x20,   %1
    movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      16
    putc    %2
    addq    1,      %1,     %1
    jmp     -20
    halt    0x41
    .string "hello, world!\n"
```

```
#TEXT 4
0x0000000000000000: 10 10 00 20 #     loadz 32, %1
0x0000000000000004: 20 21 00 00 #     movzbq (%1), %2
0x0000000000000008: 14 02 00 00 #     subq 0, %2, %0
0x000000000000000C: 04 00 00 04 #     jz 16
0x0000000000000010: 30 20 00 00 #     putc %2
0x0000000000000014: 12 11 00 01 #     addq 1, %1, %1
0x0000000000000018: 05 FF FF FB #     jmp -20
0x000000000000001C: 01 41 00 00 #     halt 65
0x0000000000000020: 68 65 6C 6C
0x0000000000000024: 6F 2C 20 77
0x0000000000000028: 6F 72 6C 64
0x000000000000002C: 21 0A 00 00 #     .string "hello, world!\n"
#DATA 1
#BSS 1 0
#SYMTAB
```

```
    loadz   S,      %1
F   movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      H
    putc    %2
    addq    1,      %1,     %1
    jmp     F
H   halt    0x41
S   .string "hello, world!\n"
```

```
    loadz   S,      %1
F   movzbq  (%1),   %2
    subq    0,      %2,     %0
    jz      H
    putc    %2
    addq    1,      %1,     %1
    jmp     F
H   halt    0x41
S   .string "hello, world!\n"
```

```
#TEXT 4
0x0000000000000000: 10 10 00 20 #     loadz S, %1
0x0000000000000004: 20 21 00 00 #     movzbq (%1), %2
0x0000000000000008: 14 02 00 00 #     subq 0, %2, %0
0x000000000000000C: 04 00 00 04 #     jz H
0x0000000000000010: 30 20 00 00 #     putc %2
0x0000000000000014: 12 11 00 01 #     addq 1, %1, %1
0x0000000000000018: 05 FF FF FB #     jmp F
0x000000000000001C: 01 41 00 00 #     halt 65
0x0000000000000020: 68 65 6C 6C
0x0000000000000024: 6F 2C 20 77
0x0000000000000028: 6F 72 6C 64
0x000000000000002C: 21 0A 00 00 #     .string "hello, world!\n"
#DATA 1
#BSS 1 0
#SYMTAB
t S                         0x0000000000000020
t F                         0x0000000000000004
t H                         0x000000000000001C
#FIXUPS
text 0x0000000000000000 12 20 absolute [text]+32
```

```
    loadz   S,      %p
F   movzbq  (%p),   %ch
    subq    0,      %ch,    %0
    jz      H
    putc    %2
    addq    1,      %p,     %p
    jmp     F
H   halt    0x41
S   .string "hello, world!\n"

    .equ    p,      1
    .equ    ch,     2
```

```
#TEXT 4
0x0000000000000000: 10 10 00 20 #     loadz S, %p
0x0000000000000004: 20 21 00 00 #     movzbq (%p), %ch
0x0000000000000008: 14 02 00 00 #     subq 0, %ch, %0
0x000000000000000C: 04 00 00 04 #     jz H
0x0000000000000010: 30 20 00 00 #     putc %2
0x0000000000000014: 12 11 00 01 #     addq 1, %p, %p
0x0000000000000018: 05 FF FF FB #     jmp F
0x000000000000001C: 01 41 00 00 #     halt 65
0x0000000000000020: 68 65 6C 6C
0x0000000000000024: 6F 2C 20 77
0x0000000000000028: 6F 72 6C 64
0x000000000000002C: 21 0A 00 00 #     .string "hello, world!\n"
#DATA 1
#BSS 1 0
#SYMTAB
t S                         0x0000000000000020
a p                         0x0000000000000001
t F                         0x0000000000000004
a ch                        0x0000000000000002
t H                         0x000000000000001C
#FIXUPS
text 0x0000000000000000 12 20 absolute [text]+32
```

```
    .data
string:
    .string "hello, world!\n"


    .equ    p,      1
    .equ    ch,     2

    .text
    loadz   string, %p
fetch:
    movzbq  (%p),   %ch
    subq    0,      %ch,    %0
    jz      halt
    putc    %2
    addq    1,      %p,     %p
    jmp     fetch
halt:
    halt    0x41
```

```
#TEXT 4
0x0000000000000000: 10 10 00 20 #     loadz string, %p
0x0000000000000004: 20 21 00 00 #     movzbq (%p), %ch
0x0000000000000008: 14 02 00 00 #     subq 0, %ch, %0
0x000000000000000C: 04 00 00 04 #     jz halt
0x0000000000000010: 30 20 00 00 #     putc %2
0x0000000000000014: 12 11 00 01 #     addq 1, %p, %p
0x0000000000000018: 05 FF FF FB #     jmp fetch
0x000000000000001C: 01 41 00 00 #     halt 65
#DATA 1
0x0000000000000020: 68 65 6C 6C 6F 2C 20 77 6F
                    72 6C 64 21 0A 00
                    #     .string "hello, world!\n"
#BSS 1 0
#SYMTAB
d string                0x0000000000000000
a p                     0x0000000000000001
a ch                    0x0000000000000002
t fetch                 0x0000000000000004
t halt                  0x000000000000001C
#FIXUPS
text 0x0000000000000000 12 20 absolute [data]+0
```