Useful instructions

- mov – moves data between registers and memory
  - `mov $1,%eax   # move 1 to register eax`
  - `n: .int 123   # label n points to an integer`
    `# variable`
    `mov n,%eax    # move value of the variable to eax`
  - `mov %eax,%ebx # copy the value in eax to ebx`
- push/pop – stack manipulation
  - Useful when we need to store data for later and we cannot use registers for that
  - `push %eax    # push content of eax to the stack`
  - `pop %ebx     # pop a value from the stack to ebx`

Useful instructions 2

- add – adds two operands
  - add $2,%eax # eax = eax + 2
  - add %eax,%ebx # ebx = ebx + eax
- sub – subtracts two operands
  - sub $2,%eax # eax = eax – 2

Useful instructions 3

- call – calls a subroutine
- ret – returns from a subroutine to the caller

```
plusone:
     add $1,%eax
     ret
main:
      mov $12,%eax
      call plusone
      ...
```

Useful instructions 4

- div – integer division (not a simple instruction)
  http://x86.renejeschke.de/html/file_module_x86_id_72.html
  - 8 bit operand: ax divided by the operand
    result: al = ax / operand, ah = ax % operand
    - mov $42,%ax
      mov $12,%bl
      div %bl # al = 42/12 = 3
  - 16 bit operand: dx:ax divided by the operand
    result: ax = dx:ax / operand, dx = dx:ax % operand
    - mov $0x1,%dx
      mov $0x2345,%ax
      mov $10,%bx
      div %bx # ax = 0x12345 / 10
  - ...

Useful instructions 5

- cmp – compare two values
  - cmp $2,%eax # compare eax with 2 and set eflags register
  - je label # jump to the label if eax was **equal** to 2
  - jl label # jump if eax was **less**
  - jg label # jump if eax was **greater**
  - jlelabel # jump if **less or equal**
  - jge label # jump if **greater or equal**
- Example**:**
  -

```
    cmp $0x30,%al
    jl nodigit
    cmp %0x39,%al
    jg nodigit
digit:
    ... do something ...
nodigit:
    ... handle error
```

Extended assembler

```
// Compile with gcc -m32 -O2 -Wall ...
#include <stdio.h>
int main()
{
    void *stack_ptr;
    asm volatile ("mov %%esp,%0;" : "=g" (stack_ptr));
    printf("Value of ESP register is %p\n", stack_ptr);
    return 0;
}
```

- Allows using C expressions in assembler instructions
- Programmer writes “instruction templates”
- Compiler replaces parameters (%0 above) with real operands (registers, memory references, ...)
- Compiler does not try to understand the asm code! Programmer has to tell what is the effect of the assembler.

Extended assembler syntax

```
#include <stdio.h>

int main(int argc, char *argv[])
{
    int result, op1 = 4, op2 = 2;
    asm volatile (
                "mov %1,%0;"
               "add %2,%0;"
               : "=r" (result)
               : "r" (op1), "r" (op2)
               : "cc");  // flags register (condition codes) is modified
    printf("result = %d\n", result);
    return 0;
}
```

Extended assembler syntax:

```
asm ( assembler template
      : output operands  /* optional*/
      : input operands   /* optional*/
      : clobber list     /* optional*/
     );
```

The syntax of operands after ":" is:

```
  <constraint> (<C expression>), ....
```

https://gcc.gnu.org/onlinedocs/gcc/Extended-Asm.html

Compiles into (objdump -d ...):

```
...
 80482c0:     ba 02 00 00 00          mov    $0x2,%edx
 80482c5:     b8 04 00 00 00          mov    $0x4,%eax
 80482ca:     89 c0                   mov    %eax,%eax
 80482cc:     01 d0                   add    %edx,%eax
...
```

Extended assembler constraints

- Tell the compiler which registers or other operands are allowed in instructions given in the template
  - https://gcc.gnu.org/onlinedocs/gcc/Constraints.html
  - Generic constraints
    - **“g” – anything**
    - **“r” – register:**
      asm volatile (“mov %0,%%eax” :: “r” (var) : “eax”) → mov %ebx,%eax
    - **“m” – memory:**
      asm volatile (“mov %0,%%eax” :: “m” (var) : “eax”) → mov var,%eax
    - **“i” – immediate operand:**
      asm volatile (“mov %0,%%eax” :: “i” (123) : “eax”) → mov $123,%eax
  - Machine (HW) specific constraints
    - “a” – *ax register (for x86)
    - “b” – *bx register (for x86)
    - ...