

Keystone Engine Assembler



Capstone Engine
Disassembler



Unicorn EngineCPU Emulator



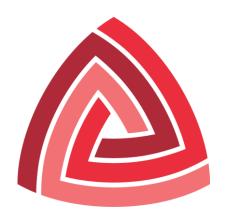
Keystone Engine

Assembler

INC ECX;
DEC EDX;



/x41/x4A



Capstone Engine

Disassembler

INC ECX;

DEC EDX;



/x41/x4A



Unicorn EngineCPU Emulator

/x41/x4A memory

ECX = 0x000009

EDX = 0x000005



ECX = 0x00000A

EDX = 0x000004



Supported Architectures

- ARM, ARM64 (AArch64/Armv8)
- Hexagon
- MIPS
- PowerPC
- Sparc
- SystemZ
- X86 (16, 32, and 64bit)



Available Bindings

- C/C++ (native implementation)
- PowerShell
- Perl
- Python
- NodeJS
- Ruby
- **Go**
- Rust
- Haskell
- OCaml

Keystone Engine Assembler

```
from keystone import *
    from struct import pack
    from binascii import hexlify
    # Separate assembly instructions by; or \n
    CODE = b"INC ECX\n" \
            "DEC EDX\n"
 8
    def main():
10
        try:
11
            # Initialize engine in X86 32-bit mode
            ks = Ks(KS\_ARCH\_X86, KS\_MODE\_32)
12
            encoding, count = ks.asm(CODE)
13
            machine_code = ""
14
            for opcode in encoding:
15
                machine_code += pack("B", opcode)
16
17
            print("Machine Code:\n%s\n" %(hexlify(machine_code)))
18
        except KsError as e:
            print("ERROR: %s" %e)
19
20
21
    if __name__ == "__main__":
        main()
22
```

root@94134cf477b3:/retriforce# python example_keystone.py
Machine Code:
414a



Supported Architectures

- ARM, ARM64 (AArch64/Armv8)
- MIPS
- PowerPC
- Sparc
- SystemZ
- XCore
- X86 (16, 32, and 64bit)



Available Bindings

- C (native implementation)
- C++
- PowerShell
- Emacs
- Haskell
- Perl
- Python
- Ruby
- C#
- NodeJS

- Java
- Go
- Ocaml
- Lua
- Rust
- Delphi
- Free Pascal
- Vala

Capstone Engine Disassembler

```
from capstone import *
    from binascii import unhexlify
    # Machine code x86 32-bit
    CODE = unhexlify("414A") # => "\x41\x4A"
    BASE_ADDRESS = 0 \times 00400526
    def main():
        try:
10
            # Initialize engine in X86 32-bit mode
11
             cap = Cs(CS\_ARCH\_X86, CS\_MODE\_32)
            for i in cap.disasm(CODE, BASE_ADDRESS):
12
13
                 print("0x%08x:\t%s\t%s" %(i.address, i.mnemonic, i.op_str))
14
        except CsError as e:
            print("ERROR: %s" %e)
15
16
    if __name__ == "__main__":
17
        main()
18
```

Capstone Engine Disassembler

```
root@5e2783a2f9a0:/retriforce# python example_capstone.py
```

 0×00400526 : inc ecx

 0×00400527 : dec edx



Supported Architectures

- ARM, ARM64 (Armv8)
- M68K
- MIPS
- Sparc
- X86 (16, 32, and 64bit)



CPU Emulator

Available Bindings

- C (native implementation)
- Visual Basic
- Perl
- Rust
- Haskell
- Ruby
- Python

- Java
- **Go**
- C#
- Delphi
- Free Pascal

```
from unicorn import *
    from unicorn.x86_const import *
    from binascii import hexlify, unhexlify
    X86_CODE32 = unhexlify("414A")
    BASE ADDRESS = 0 \times 0040059d
    def main():
        try:
            emu = Uc(UC\_ARCH\_X86, UC\_MODE\_32)
10
            emu.mem_map((BASE_ADDRESS/0 \times 1000)*0 \times 1000, 2 * 1024 * 1024)
11
            emu.mem_write(BASE_ADDRESS, X86_CODE32)
12
            emu.req_write(UC_X86_REG_ECX, 0x9)
13
            emu.reg_write(UC_X86_REG_EDX, 0x5)
14
            emu.emu_start(BASE_ADDRESS, BASE_ADDRESS + len(X86_CODE32))
15
            print("Emulation done.")
16
17
            print(">>> ECX = 0x%08x" % emu.reg_read(UC_X86_REG_ECX))
            print(">>> EDX = 0x%08x" % emu.reg_read(UC_X86_REG_EDX))
18
19
        except UcError as e:
20
             print("ERROR: %s" % e)
21
22
   if __name__ == "__main__":
23
        main()
```

```
root@cecf1e08a505:/retriforce# python example_unicorn.py
Emulation done.
>>> ECX = 0x00000000a
>>> EDX = 0x00000004
```