



***Keystone Engine***  
*Assembler*



***Capstone Engine***  
*Disassembler*



***Unicorn Engine***  
*CPU Emulator*



***Keystone Engine***  
*Assembler*

```
INC ECX;  
DEC EDX;
```



/x41/x4A



***Capstone Engine***  
*Disassembler*

```
INC ECX;  
DEC EDX;
```



/x41/x4A



***Unicorn Engine***  
*CPU Emulator*

/x41/x4A  memory

ECX = 0x000009

EDX = 0x000005



ECX = 0x00000A

EDX = 0x000004



***Keystone Engine***  
*Assembler*

## **Supported Architectures**

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



***Keystone Engine***  
*Assembler*

## **Available Bindings**

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



# Keystone Engine Assembler

```
1  from keystone import *
2  from struct import pack
3  from binascii import hexlify
4
5  # Separate assembly instructions by ; or \n
6  CODE = b"INC ECX\n" \
7         "DEC EDX\n"
8
9  def main():
10     try:
11         # Initialize engine in X86 32-bit mode
12         ks = Ks(KS_ARCH_X86, KS_MODE_32)
13         encoding, count = ks.asm(CODE)
14         machine_code = ""
15         for opcode in encoding:
16             machine_code += pack("B", opcode)
17         print("Machine Code:\n%s\n" %(hexlify(machine_code)))
18     except KsError as e:
19         print("ERROR: %s" %e)
20
21 if __name__ == "__main__":
22     main()
```



# *Keystone Engine Assembler*

```
root@94134cf477b3:/retriforce# python example_keystone.py
```

```
Machine Code:
```

```
414a
```



***Capstone Engine***  
*Disassembler*

## **Supported Architectures**

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



***Capstone Engine***  
*Disassembler*

## 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

```
1  from capstone import *
2  from binascii import unhexlify
3
4  # Machine code x86 32-bit
5  CODE = unhexlify("414A") # => "\x41\x4A"
6  BASE_ADDRESS = 0x00400526
7
8  def main():
9      try:
10         # Initialize engine in X86 32-bit mode
11         cap = Cs(CS_ARCH_X86, CS_MODE_32)
12         for i in cap.disasm(CODE, BASE_ADDRESS):
13             print("0x%08x:\t%s\t%s" %(i.address, i.mnemonic, i.op_str))
14     except CsError as e:
15         print("ERROR: %s" %e)
16
17 if __name__ == "__main__":
18     main()
```



# *Capstone Engine Disassembler*

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

```
0x00400526:      inc      ecx
```

```
0x00400527:      dec      edx
```



***Unicorn Engine***  
*CPU Emulator*

## **Supported Architectures**

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



***Unicorn Engine***  
*CPU Emulator*

## **Available Bindings**

- C (native implementation)
- Visual Basic
- Perl
- Rust
- Haskell
- Ruby
- Python
- Java
- Go
- C#
- Delphi
- Free Pascal



# Unicorn Engine CPU Emulator

```
1  from unicorn import *
2  from unicorn.x86_const import *
3  from binascii import hexlify, unhexlify
4
5  X86_CODE32 = unhexlify("414A")
6  BASE_ADDRESS = 0x0040059d
7
8  def main():
9      try:
10         emu = Uc(UC_ARCH_X86, UC_MODE_32)
11         emu.mem_map((BASE_ADDRESS/0x1000)*0x1000, 2 * 1024 * 1024)
12         emu.mem_write(BASE_ADDRESS, X86_CODE32)
13         emu.reg_write(UC_X86_REG_ECX, 0x9)
14         emu.reg_write(UC_X86_REG_EDX, 0x5)
15         emu.emu_start(BASE_ADDRESS, BASE_ADDRESS + len(X86_CODE32))
16         print("Emulation done.")
17         print(">>> ECX = 0x%08x" % emu.reg_read(UC_X86_REG_ECX))
18         print(">>> EDX = 0x%08x" % emu.reg_read(UC_X86_REG_EDX))
19     except UcError as e:
20         print("ERROR: %s" % e)
21
22 if __name__ == "__main__":
23     main()
```



# *Unicorn Engine CPU Emulator*

```
root@cecf1e08a505:/retriforce# python example_unicorn.py  
Emulation done.  
>>> ECX = 0x00000000a  
>>> EDX = 0x000000004
```