如何用Go模拟CPU

蒙卓

华为 – 2012实验室
工程师

本次演讲不代表华为公司及其立场

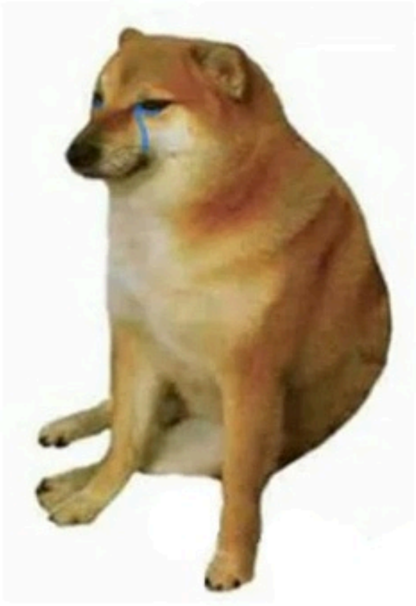
GopherChina 2021

目 录

1970年程序员
CPU 80KHz 单核
内存 64KB 手编磁芯

老娘把你送上月球

2021年程序员
CPU 2,400,000KHz 4核
内存 8,000,000KB DDR3



呜呜
App内存不足
外卖下不了单

概念太多，程序猿太苦了！

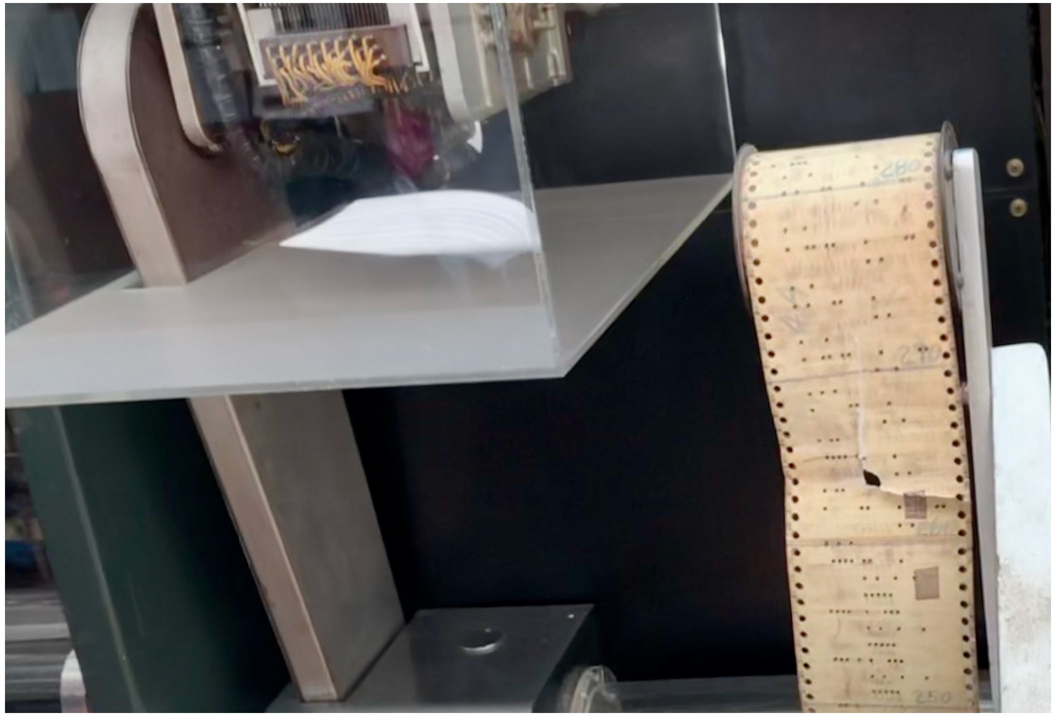
linguistics
language
like
their
about
from
development
intelligence
research
things
cognitive
director
languages
world
scientist
engineering
austin
grad
human
data
natural
machine
programming
master
founder
analytics
into
artificial
social
that
semantic
tweet
processing
search
researcher
analysis
more
candidate
apache
software
computer
developer
science
linguist
student
information
engineer
life
work
services
sentiment
knowledge
your
business
text
tweets
digital
marketing
people
passionate
distributed
learning
media
computational
with
interested
university
working
currently
graduate
python
technology
professor
twitter
tech
love
scala
based
chief
java
free
trying
open
health
systems
texas

概念是怎么来的？

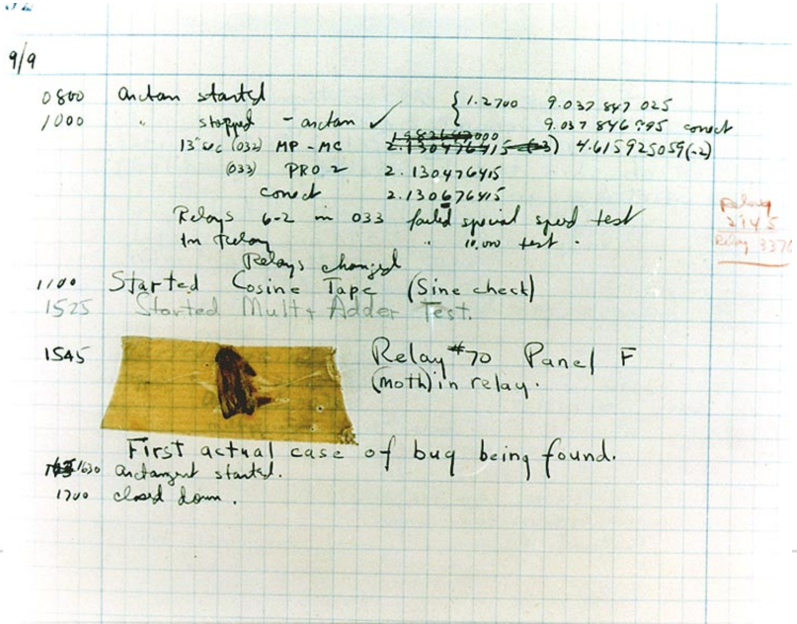

linguistics
data
natural
machine
research
from
development
intelligence
technologies
director
cognitive
things
master
founder
analytics
into
artificial
understand
work
software
services
companies
sentiment
knowledge
learning
media
scala
tech
love
working
systems
innovative
developer
tweets
text
science
linguist
information
engineer
life
interested
professor
processing
researcher
candidate
apache
analysis
more
search
manager
free
java
twitter
texas
wannabe
trying
open
chief
associate
based
news
code

计算机的演化历史

- 跟其他发明一样：一部偷懒的历史
- 继电器计算机
  - 1937年贝尔实验室：使用机械继电器
  - **从织布机来源的灵感，可使用纸带打孔的方式编写程序（patch的来源）**
- 真空管计算机
  - 1946年 ENIAC，由于减少了继电器的机械装置速度更快，但寿命短。
  - **因为真空管会发光，吸引飞蛾（bug的来源）**
- 集成电路计算机
  - 1947年 贝尔实验室：晶体管诞生，现代计算机开始得以应用

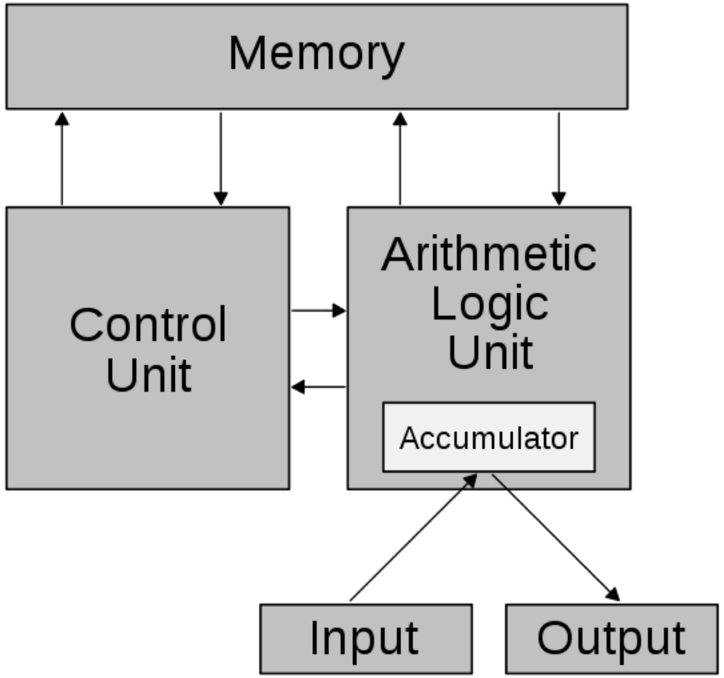


本人摄于哈佛计算机学院



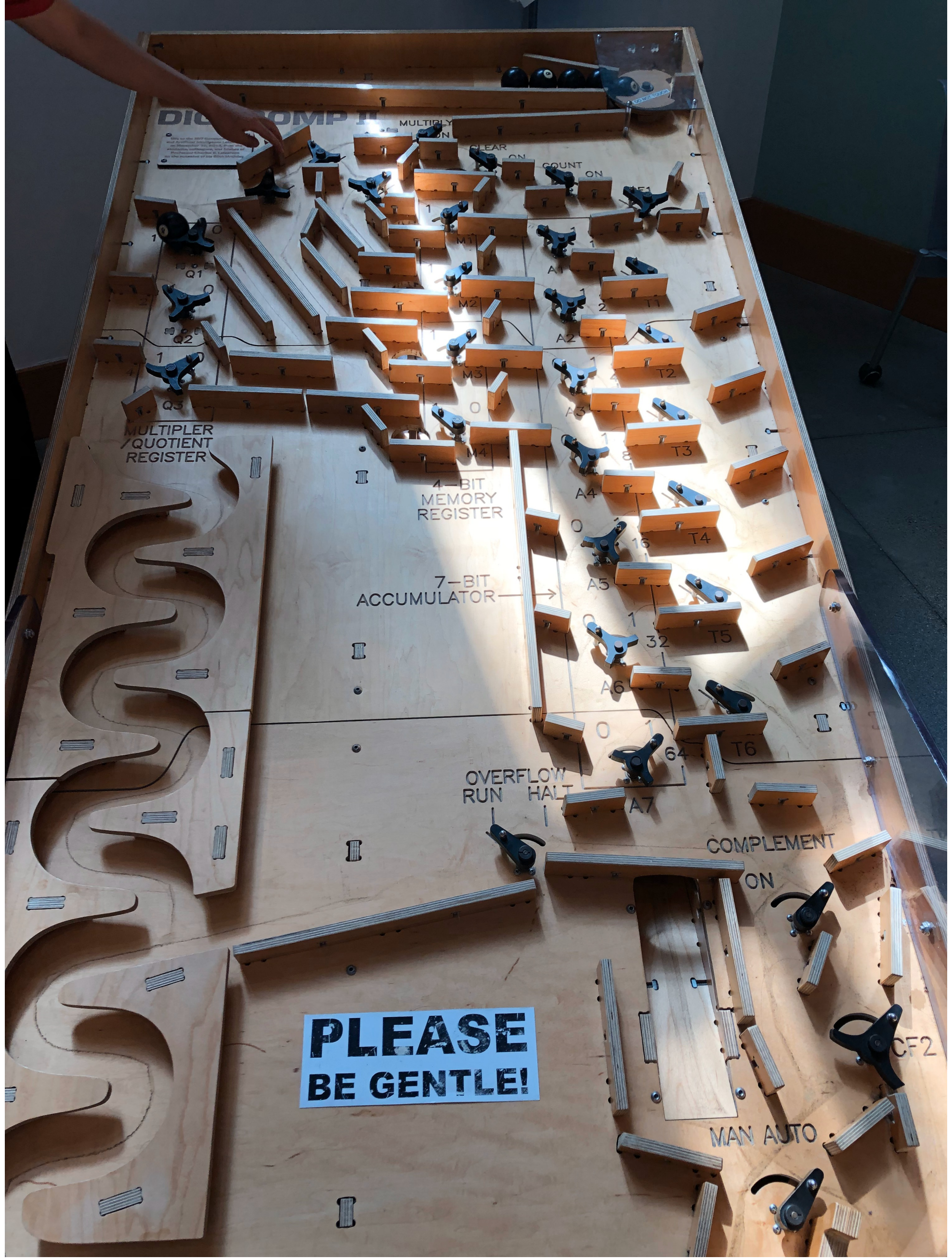
计算机的演化历史

- 因为改纸带比较麻烦
- 冯诺伊曼架构
  - 又称存储程序型计算机
  - 可在运行时改变指令
  - 指令控制数据



计算机的演化历史

- 因为改纸带比较麻烦
- 冯诺伊曼架构
  - 又称存储程序型计算机
  - 可在运行时改变指令
  - 指令控制数据
  - 用啥实现他老人家可没说（木头计算机）



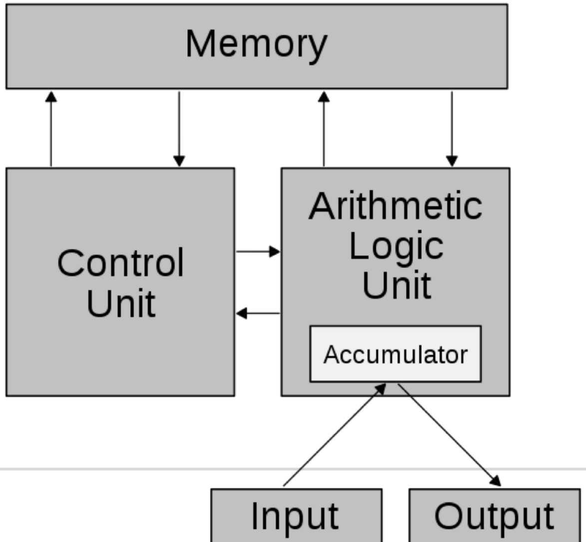
本人摄于MIT CSAI Lab

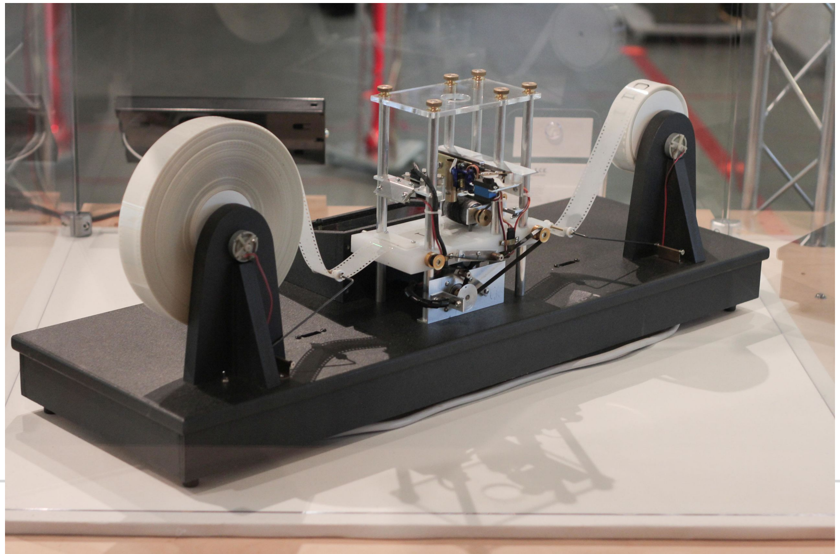
计算机的演化历史

- 现代计算机绝大部份是冯诺伊曼架构
- 所有发明（概念）都是有基础的，切忌“名词理解”
- 偷懒都是有代价的：抽象层概念
  - 纸带 vs 内存
- 现代程序员学的都是复杂概念

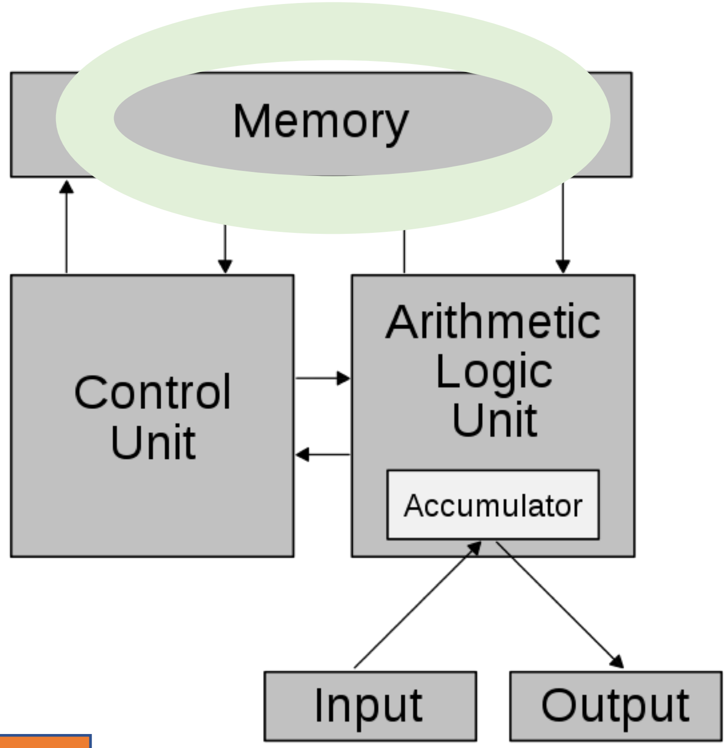
Go模拟计算机

- 如何学习计算机概念？
  - Go语言工具完备、语法简单~~（这次大会是Go所以用Go）~~
  - 最好的时代，现代计算机真的快且便宜
- CPU的Go实现：一个循环+一个大数组

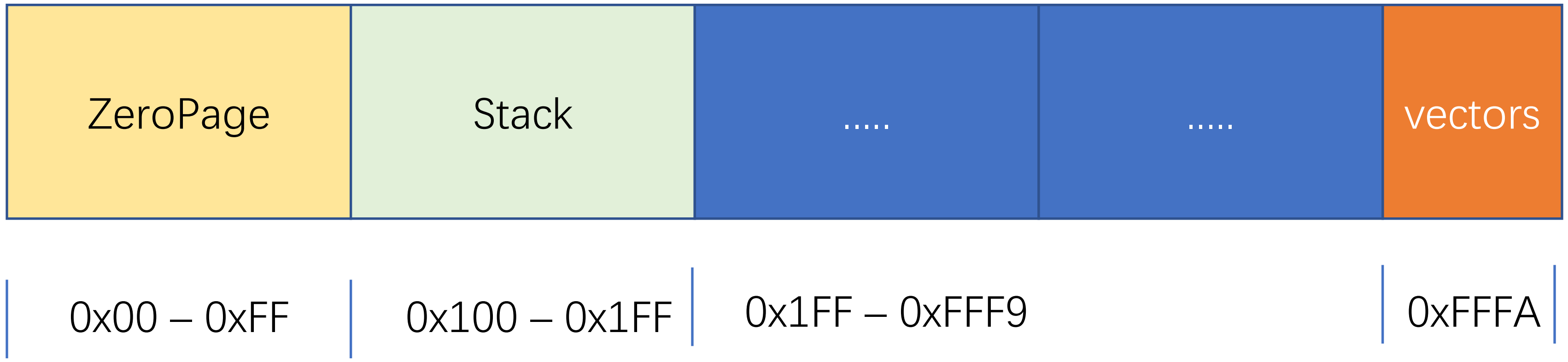




Go模拟内存



- 最简单
- 内存：超高速可改写的纸带
- 内存空间 **[65536]Byte**

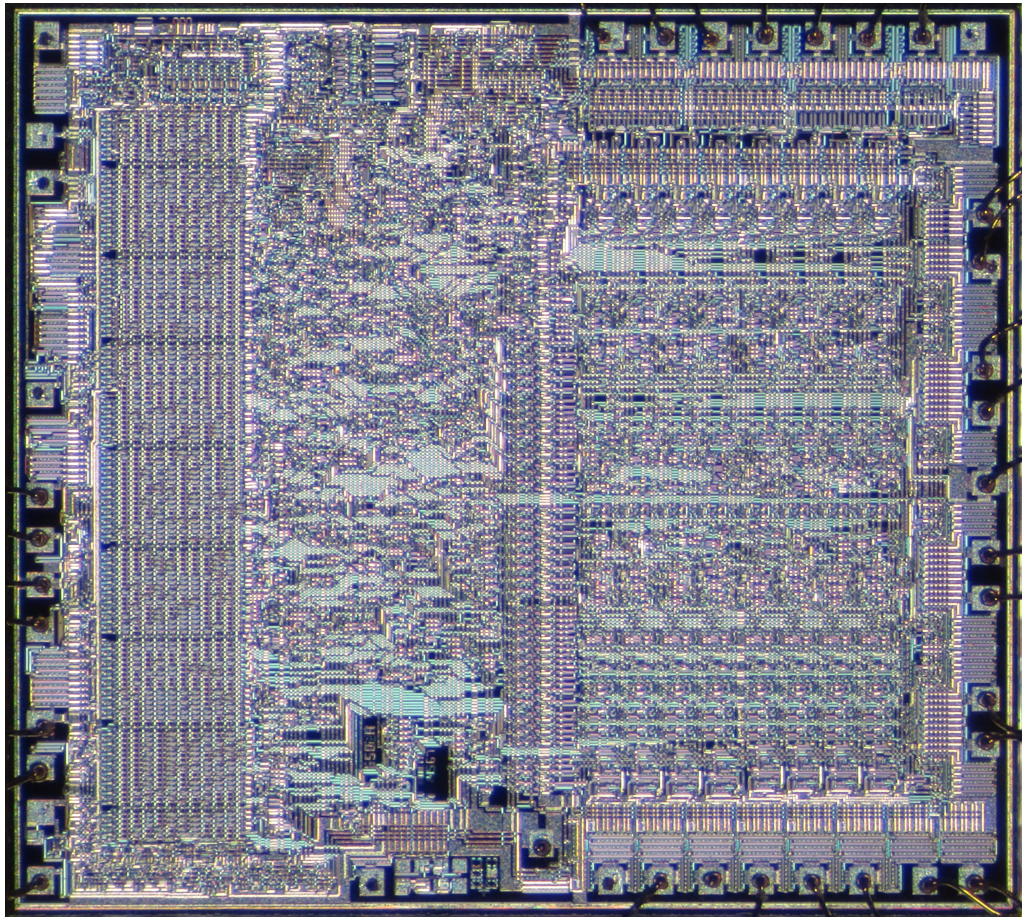


模拟CPU – MOS 6502

- 诞生于1975年
- MOS 6502应用范围广
- 资料多且易获得
- 简单、容易实现的现代CPU

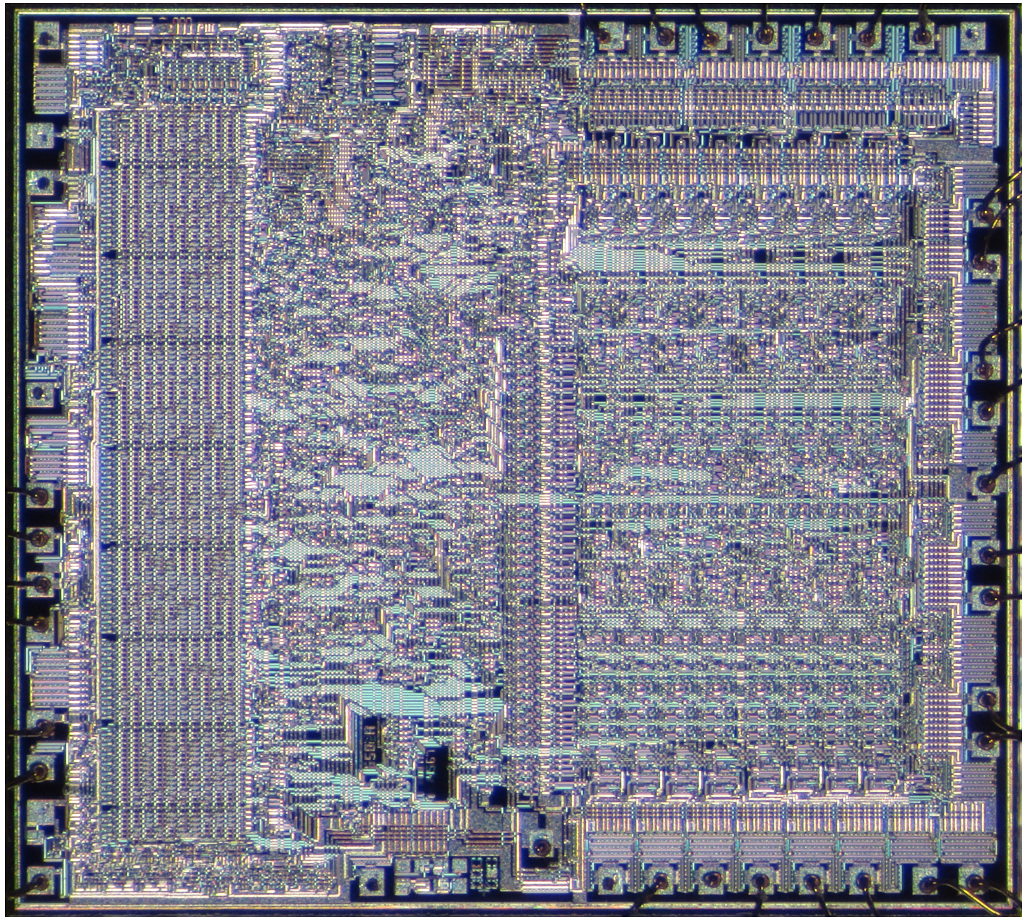
MOS 6502简介

- 8位，变长ISA（CISC）
- 寄存器
  - 1个16位程序计数器（PC）
  - 1个累加寄存器（Accumulator）
  - 2个地址索引寄存器（X，Y）
  - 1个状态寄存器（PS）
  - 1个栈寄存器（SP）
- 中断（NMI，IRQ）



MOS 6502简介

- 8位，变长ISA（CISC）
- 寄存器
  - 1个16位程序计数器（PC）
  - 1个累加寄存器（Accumulator）
  - 2个地址索引寄存器（X，Y）
  - 1个状态寄存器（PS）
  - 1个栈寄存器（SP）
- 中断（NMI，IRQ）
- 不要害怕，我们一步步实现



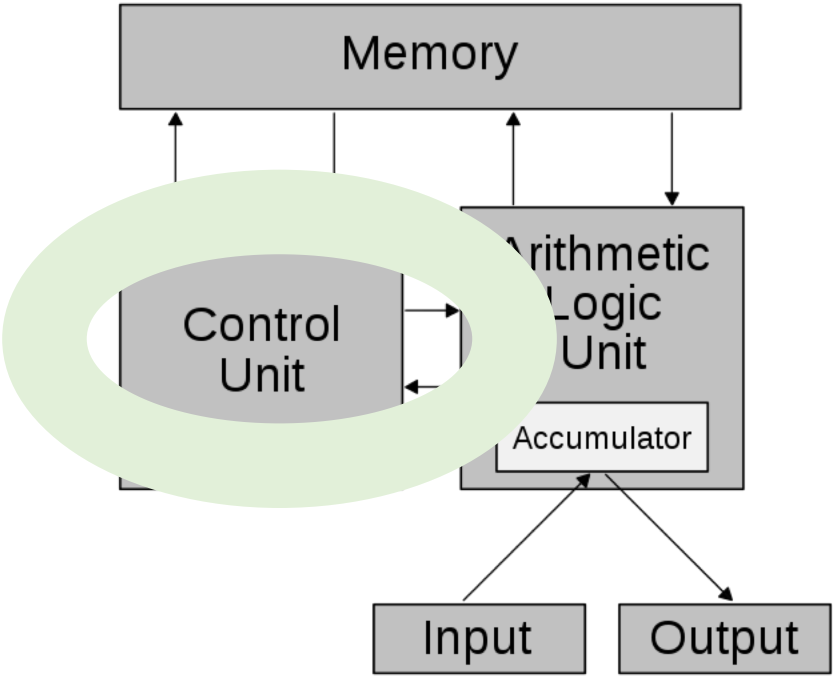
Go模拟6502控制单元

- 内存 -> 读取 -> 指令
- 16位Program Counter（PC）
- 用Go模拟： PC uint16
- 6502 支持 NOP指令（0xEA）
- 能执行NOP，就实现了控制单元

```go
type CPU struct {
  PC uint16
}

func (c *CPU) Run() {
  for {
      op := mem[c.PC]
      ins := c.Decode(op)
      c.Execute(ins)
  }
}

func (c *CPU) Decode(op uint8) (ins Instrction) {
  if op == 0xEA {
    return NOP
  }
}

func (c *CPU) Execute(ins Instruction) {
  switch ins {
    case NOP:
        c.PC += 1
  }
}
```

Go模拟6502控制单元

- 指令结构（instruction）
  - 操作码（valid opcode）
  - 寻址模式 （address mode）
    - 从内存哪里读数据
    - Implied
    - Accumulator
    - Immediate
    - Absolute
    - Zeropage, X, Y
    - Indirected, X, Y
  - 指令长度（instruction length）
  - 指令周期（cycle）
  - 1-3个字节



- NOP：啥都不做指令
  - 操作码：0xEA
  - 寻址模式：Implied（默认）
  - 指令长度：1
  - 指令周期：1

Go模拟6502控制单元

- 6502支持的指令：151个

6502 Instruction Set

HI	*LO-NIBBLE*															
	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F
0-	BRK impl	ORA X,ind				ORA zpg	ASL zpg		PHP impl	ORA #	ASL A			ORA abs	ASL abs	
1-	BPL rel	ORA ind,Y				ORA zpg,X	ASL zpg,X		CLC impl	ORA abs,Y				ORA abs,X	ASL abs,X	
2-	JSR abs	AND X,ind			BIT zpg	AND zpg	ROL zpg		PLP impl	AND #	ROL A		BIT abs	AND abs	ROL abs	
3-	BMI rel	AND ind,Y				AND zpg,X	ROL zpg,X		SEC impl	AND abs,Y				AND abs,X	ROL abs,X	
4-	RTI impl	EOR X,ind				EOR zpg	LSR zpg		PHA impl	EOR #	LSR A		JMP abs	EOR abs	LSR abs	
5-	BVC rel	EOR ind,Y				EOR zpg,X	LSR zpg,X		CLI impl	EOR abs,Y				EOR abs,X	LSR abs,X	
6-	RTS impl	ADC X,ind				ADC zpg	ROR zpg		PLA impl	ADC #	ROR A		JMP ind	ADC abs	ROR abs	
7-	BVS rel	ADC ind,Y				ADC zpg,X	ROR zpg,X		SEI impl	ADC abs,Y				ADC abs,X	ROR abs,X	
8-		STA X,ind			STY zpg	STA zpg	STX zpg		DEY impl		TXA impl		STY abs	STA abs	STX abs	
9-	BCC rel	STA ind,Y			STY zpg,X	STA zpg,X	STX zpg,Y		TYA impl	STA abs,Y	TXS impl			STA abs,X		
A-	LDY #	LDA X,ind	LDX #		LDY zpg	LDA zpg	LDX zpg		TAY impl	LDA #	TAX impl		LDY abs	LDA abs	LDX abs	
B-	BCS rel	LDA ind,Y			LDY zpg,X	LDA zpg,X	LDX zpg,Y		CLV impl	LDA abs,Y	TSX impl		LDY abs,X	LDA abs,X	LDX abs,Y	
C-	CPY #	CMP X,ind			CPY zpg	CMP zpg	DEC zpg		INY impl	CMP #	DEX impl		CPY abs	CMP abs	DEC abs	
D-	BNE rel	CMP ind,Y				CMP zpg,X	DEC zpg,X		CLD impl	CMP abs,Y				CMP abs,X	DEC abs,X	
E-	CPX #	SBC X,ind			CPX zpg	SBC zpg	INC zpg		INX impl	SBC #	NOP impl		CPX abs	SBC abs	INC abs	
F-	BEQ rel	SBC ind,Y				SBC zpg,X	INC zpg,X		SED impl	SBC abs,Y				SBC abs,X	INC abs,X	

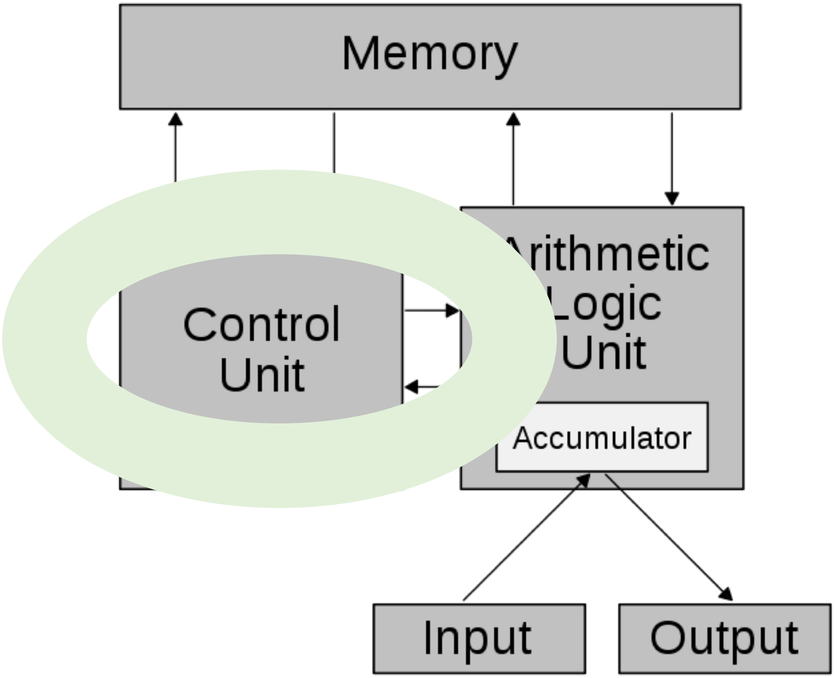
Go模拟6502控制单元

- 6502支持的指令：151个
- html解析器生成对应的指令 https://www.masswerk.at/6502/6502_instruction_set.html

```go
// Code generated by "python gen.py && go fmt"; DO NOT EDIT.

    package ins
    const (
    ADCImmediate = 0x69
ADCZeroPage = 0x65
ADCZeroPageX = 0x75
ADCAbsolute = 0x6D
ADCAbsoluteX = 0x7D
ADCAbsoluteY = 0x79
ADCIndirectX = 0x61
ADCIndirectY = 0x71
```

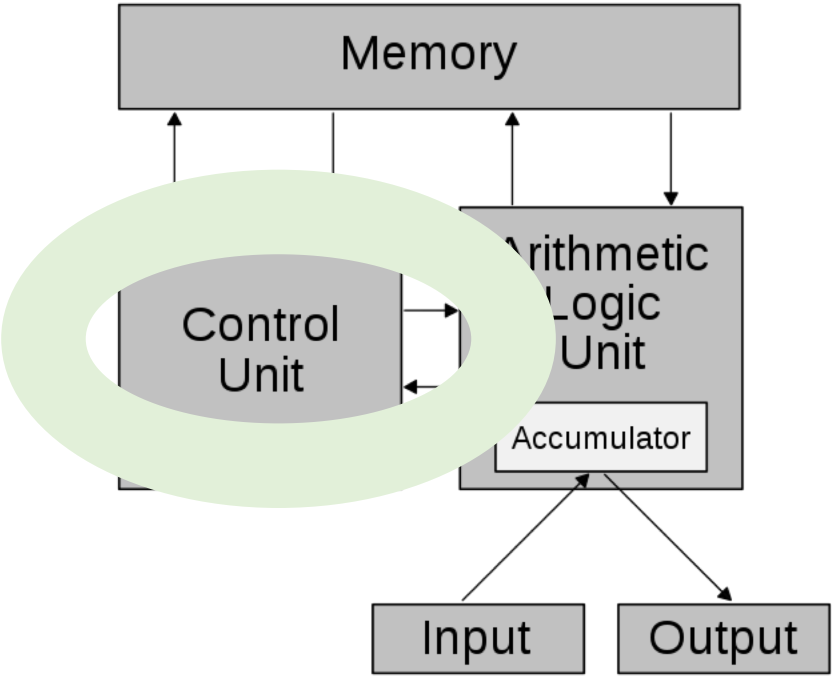
Go模拟6502控制单元

- PC根据指令跳转（Branch）
  - 有条件跳转（IF 语句，状态寄存器 uint8）
  - 无条件跳转 （NOP算其中一种）
  - 按内存里的地址跳转地址（指针/向量）

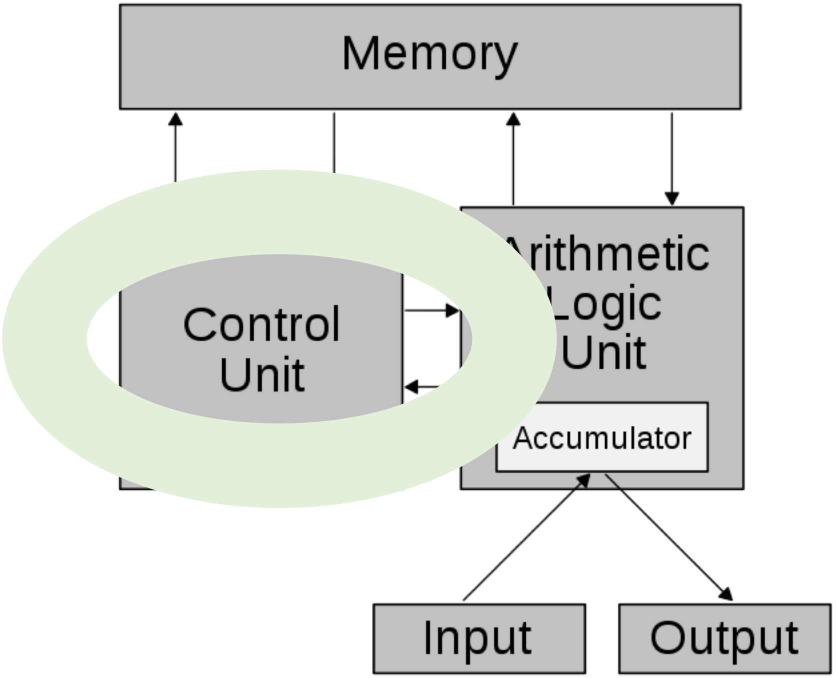


Go模拟6502控制单元

- PC根据指令跳转（Branch）
  - 有条件跳转（IF 语句，状态寄存器 uint8）
  - 无条件跳转 （NOP算其中一种）
  - 按内存里的地址跳转地址（指针/向量）
- SP与栈空间
  - 栈是人为规定的内存范围，有特殊的指令控制
  - SP（Stack Pointer）跟PC差不多，就是一个uint8
  - SP跟常见的栈指针一样，压栈SP则减掉对应字长



Go模拟6502控制单元



- PC根据指令跳转（Branch）
  - 有条件跳转（IF 语句，状态寄存器 uint8）
  - 无条件跳转 （NOP算其中一种）
  - 按内存里的地址跳转地址（指针/向量）
- SP与栈空间
  - 栈是人为规定的内存范围，有特殊的指令控制
  - SP（Stack Pointer）跟PC差不多，就是一个uint8
  - SP跟常见的栈指针一样，压栈SP则减掉对应字长
- 中断与向量
  - IRQ = 可忽略的中断 = 读取0xFFFC里的绝对地址，PC跳到那里
  - NMI = 不可忽略的中断 = 读取0xFFFE里的绝对地址，PC跳到那里
  - Reset = 重置 = 读取0xFFFA 里的地址，同上

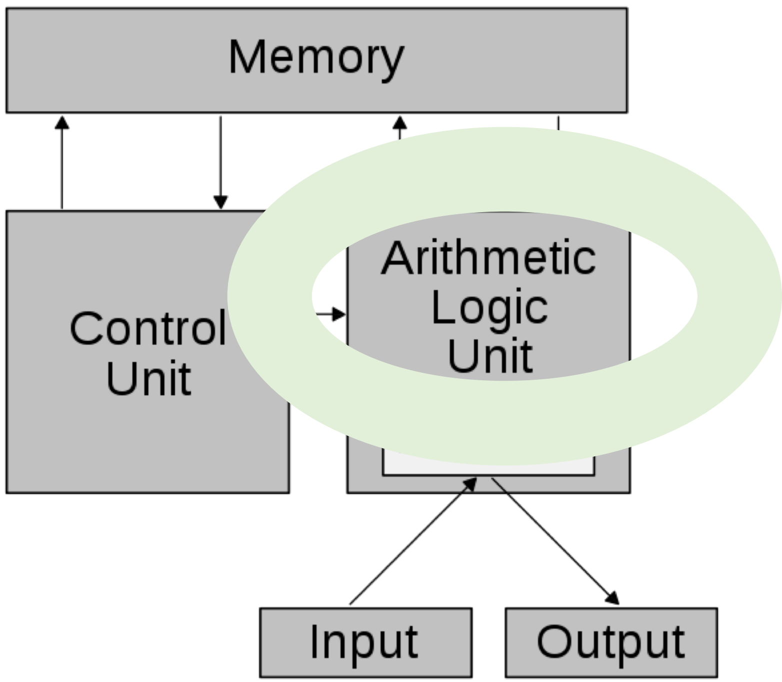
Go模拟6502控制单元

```go
for {
        select {
        case <-c.NMI:
                c.pushWordToStack(c.PC)
                c.pushByteToStack(c.PS | FlagUnused)
                c.PC = c.Mem.ReadWord(c.nmiVec)
                c.PS |= FlagIRQDisable
        case <-c.Reset:
                c.ResetF(m)
        default:
        }

        if c.IRQ != 0 && c.PS&FlagIRQDisable == 0 {
                c.pushWordToStack(c.PC)
                c.pushByteToStack(c.PS | FlagUnused)
                c.PC = c.Mem.ReadWord(c.irqVec)
                c.PS |= FlagIRQDisable
        }
```

全部控制单元
总共300行左右
不难吧

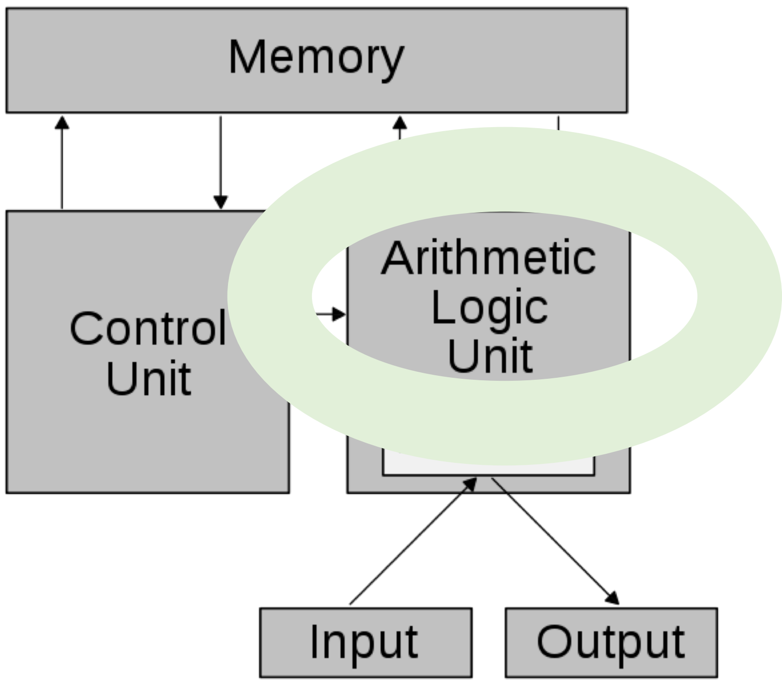
Go模拟6502计算单元

- 计算指令类型：
  - 只能在累加器（Accumulator）上操作
  - X86 -> AX 就是历史的遗留
  - Uint8 加减运算
  - Uint8 位（布尔）运算
- 影响PS（处理器状态寄存器 uint8）



Go模拟6502计算单元

- 计算指令类型：
  - 只能在累加器（寄存器A）上操作
  - Uint8 加减运算
  - Uint8 位（布尔）运算
- 影响PS（运行状态寄存器 uint8）



- ADC：带进位加法
- A = A+运算数（operand）+Carry
- 如果溢出则PS |= FlagCarry

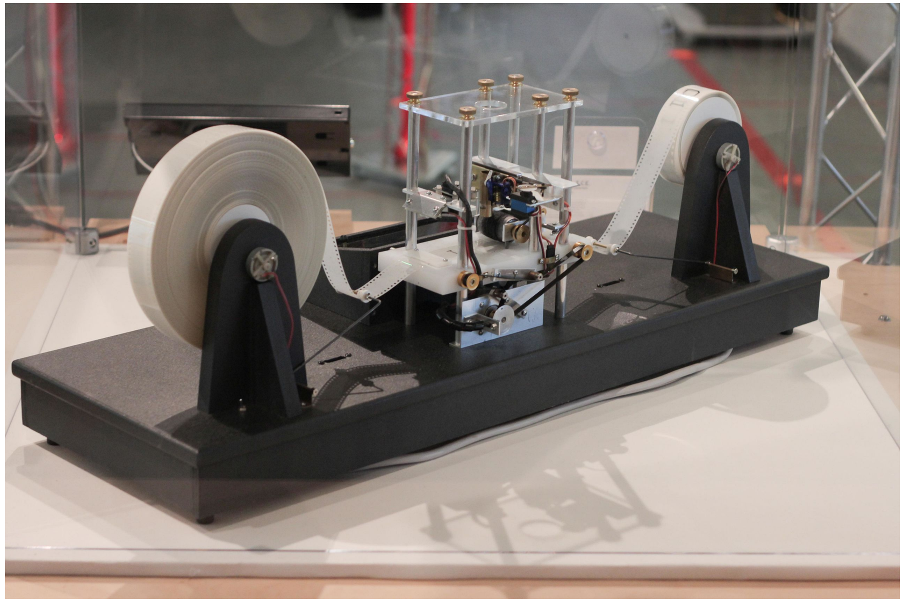
Go模拟6502计算单元

- 指令类型：
  - Uint8 加减运算
  - Uint8 位（布尔）运算
- 影响PS（运行状态寄存器）
- 加法指令
  - 也就30行
  - 取补码就减法了（省电路又省钱）
  - 补码，人造的概念，最高位是1为负

```go
// Arithmetic
case ins.ADC, ins.SBC:
	oper, cycles = c.getOper(i.Mode)
	if i.Name == ins.SBC {
		oper = ^oper
	}

	if c.PS&FlagDecimalMode != 0 {
		err = fmt.Errorf("no decimal mode")
		return
	}

	sameSigned := (c.AC^oper)&FlagNegtive == 0
	sum := uint16(c.AC)
	sum += uint16(oper)
	if c.PS&FlagCarry != 0 {
		sum += 1
	}
	c.AC = uint8(sum & 0xff)
	c.setZN(c.AC)
	if sum > 0xff {
		c.PS |= FlagCarry
	} else {
		c.PS &= ^FlagCarry
	}

	if sameSigned && ((c.AC^oper)&FlagNegtive != 0) {
		c.PS |= FlagOverflow
	} else {
		c.PS &= ^FlagOverflow
	}
```

Go模拟6502

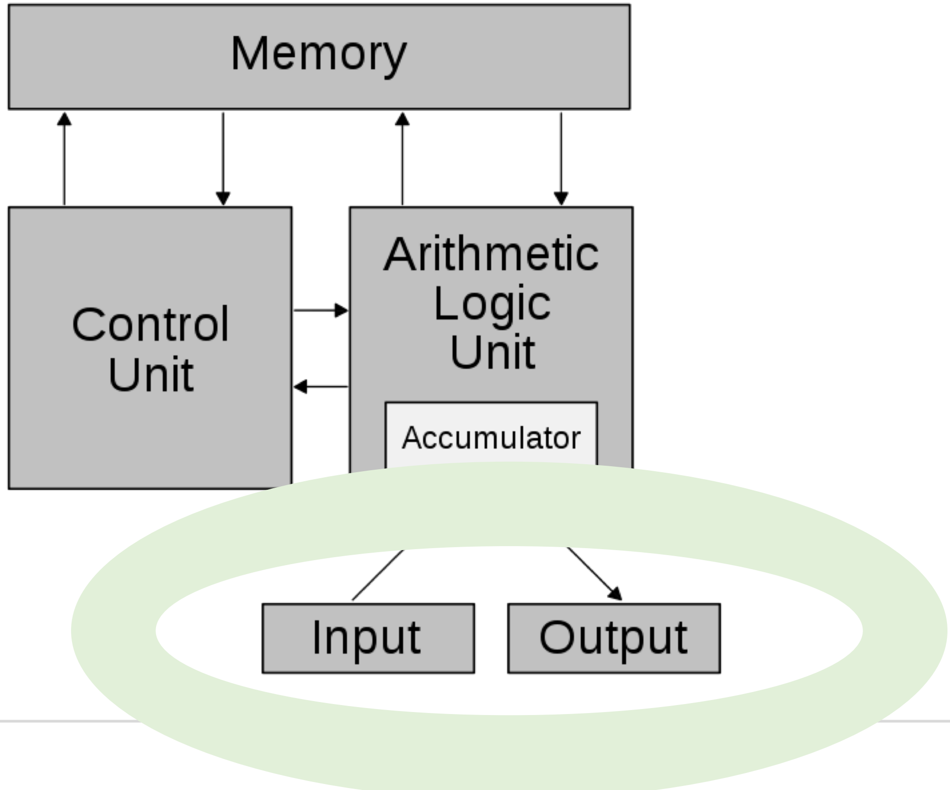
- 小结
  - 冯诺伊曼架构 约等于 有限状态机
  - 完成后实际就是6502 虚拟机（virtual machine）
  - 用Go实现全部合法指令并测试通过也就1000行左右



Go模拟6502

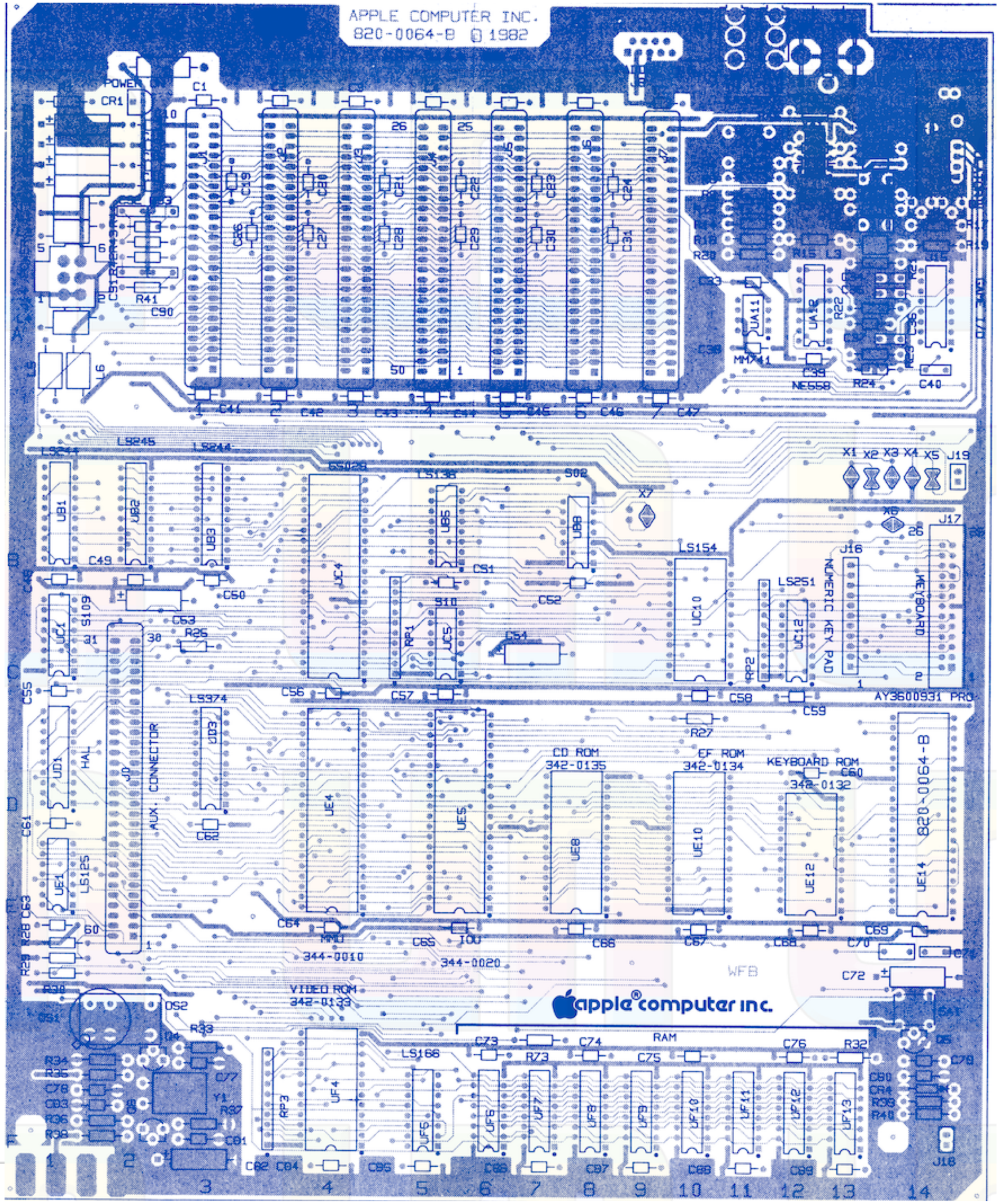
- 小结
  - 冯诺伊曼架构 约等于 有限状态机
  - 完成后实际就是6502 虚拟机（virtual machine）

说了这么多CPU
那电脑咋工作的？



Apple II

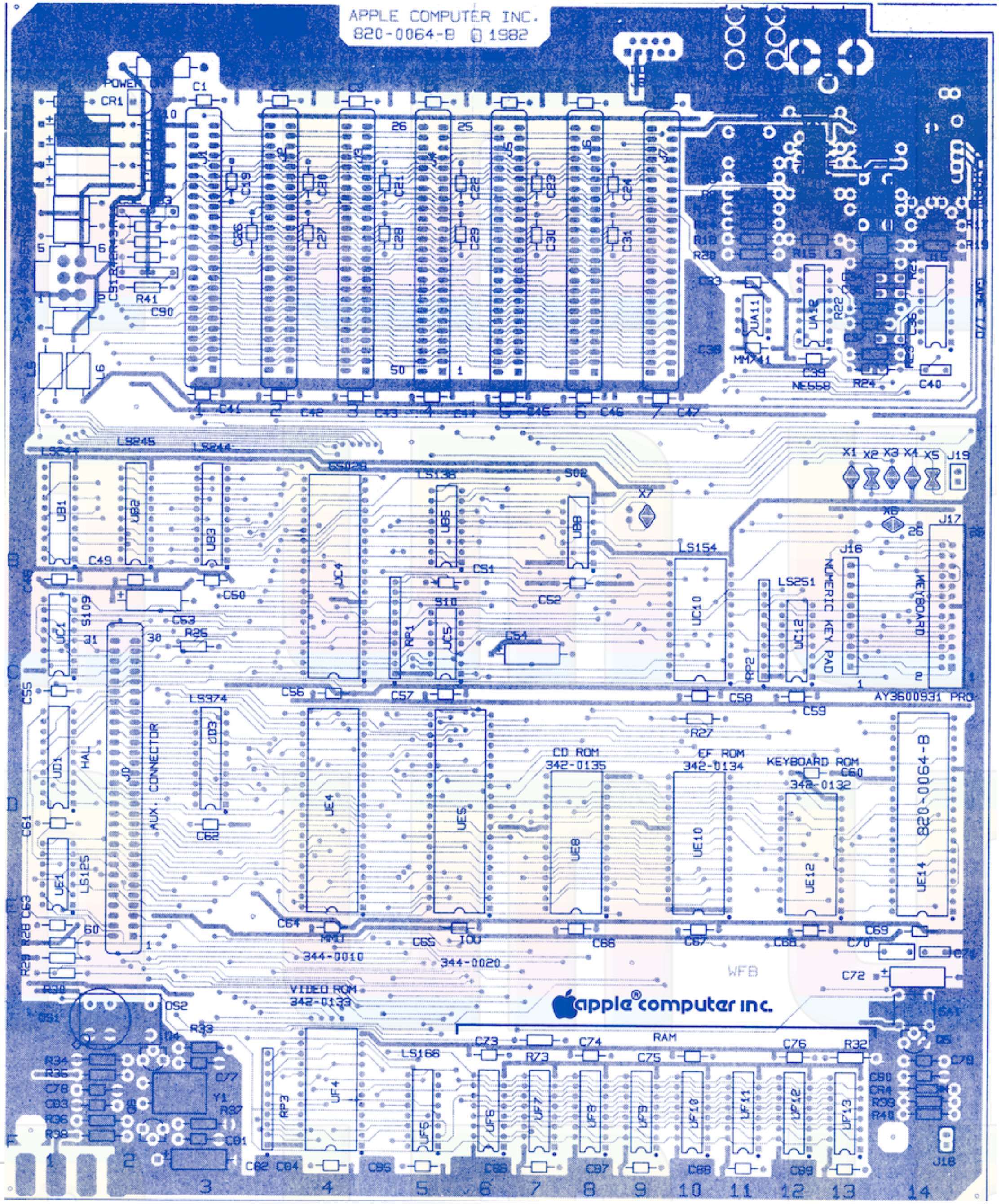
- 规格
  - CPU （MOS 6502）
  - 内存（64KiB）
  - 字符显示（LoRes 40x24）
  - 像素显示（280×192像素）
  - 输入（内置键盘）
  - 存储（磁带/5.25英寸）
- 早期内存非常昂贵
  - 4KiB = 5,543 USD (2020)
  - 64KiB = 11,266 USD (2020)



Apple II

- 规格
  - CPU （MOS 6502）
  - 内存（64KiB）
  - 字符显示（LoRes 40x24）
  - 像素显示（280×192像素）
  - 输入（内置键盘）
  - 存储（磁带/5.25英寸）

- 早期内存非常昂贵
  - 4KiB = 5,543 USD (2020)
  - 64KiB = 11,266 USD (2020)
  - 真果粉应该搞一套Apple II



Apple II

- 规格
  - CPU （MOS 6502）
  - 内存（64KiB）
  - 字符显示（LoRes 40x24）
  - 像素显示（280×192像素）
  - 输入（内置键盘）
  - 存储（磁带/5.25英寸）

- 早期内存非常昂贵
  - 4KiB = 5,543 USD (2020)
  - 64KiB = 11,266 USD (2020)
  - 真果粉应该搞一套Apple II

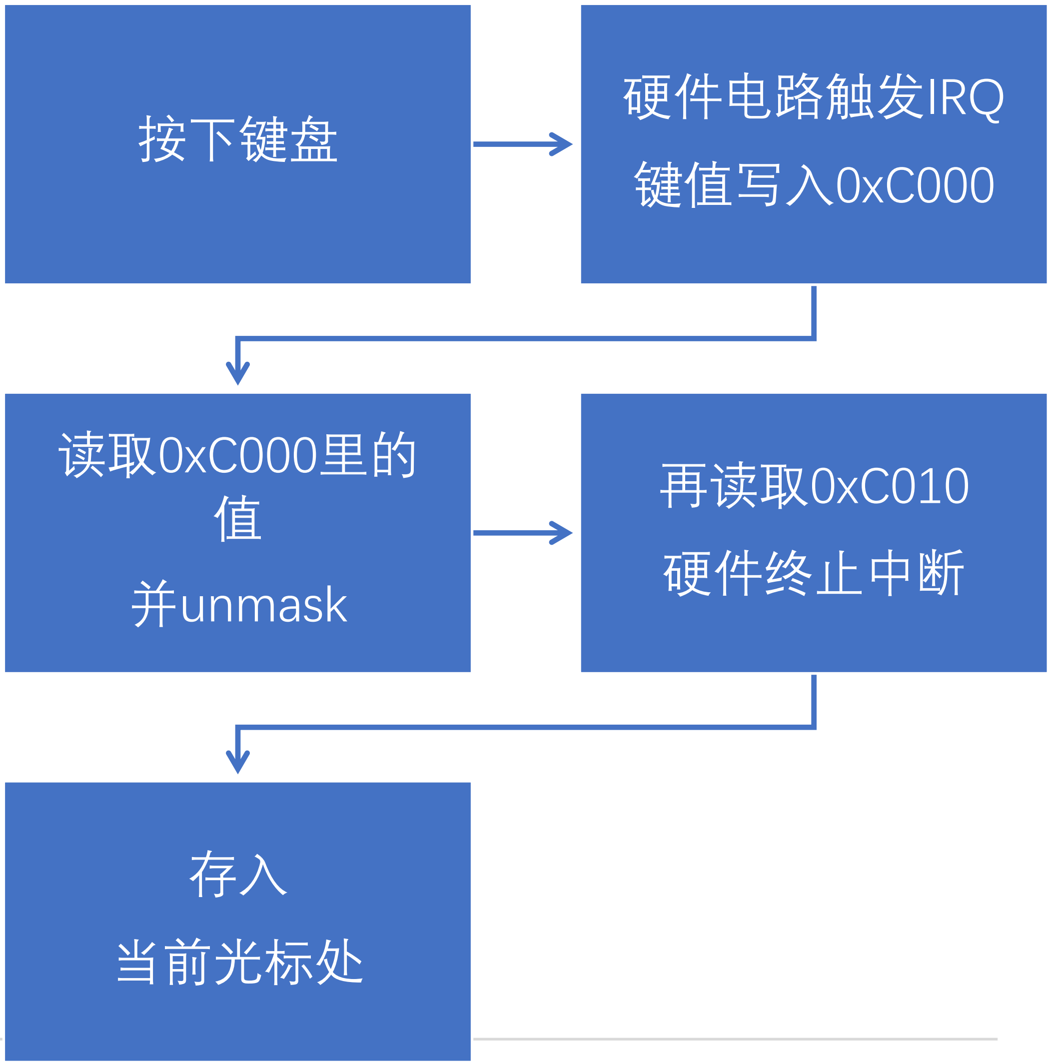
Go模拟Apple II

- 模拟Apple II 规格
  - CPU （MOS 6502）
  - 内存（64KiB） （立省1万刀）
  - 显示（LoRes 40x24） （实现起来容易）
  - 输入（内置键盘） （手头的就行）
  - ~~存储（磁带/5.25英寸）~~

Go模拟Apple II

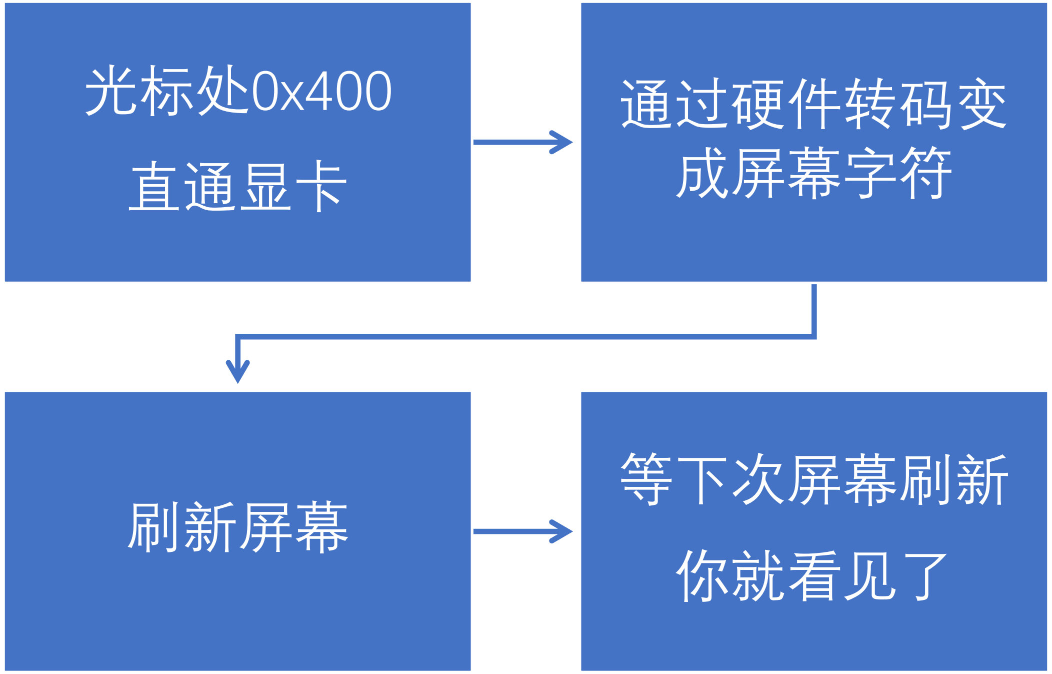
- 又双叒叕是一个大循环
  - Reset -> 启动检查程序
  - 检查完成把电脑规格写入内存指定区域
  - 一直循环monitor 函数
  - 直到有NMI/IRQ中断
- 还是不懂……
  - 就来个键盘敲下，到显示的全流程？

Go模拟Apple II

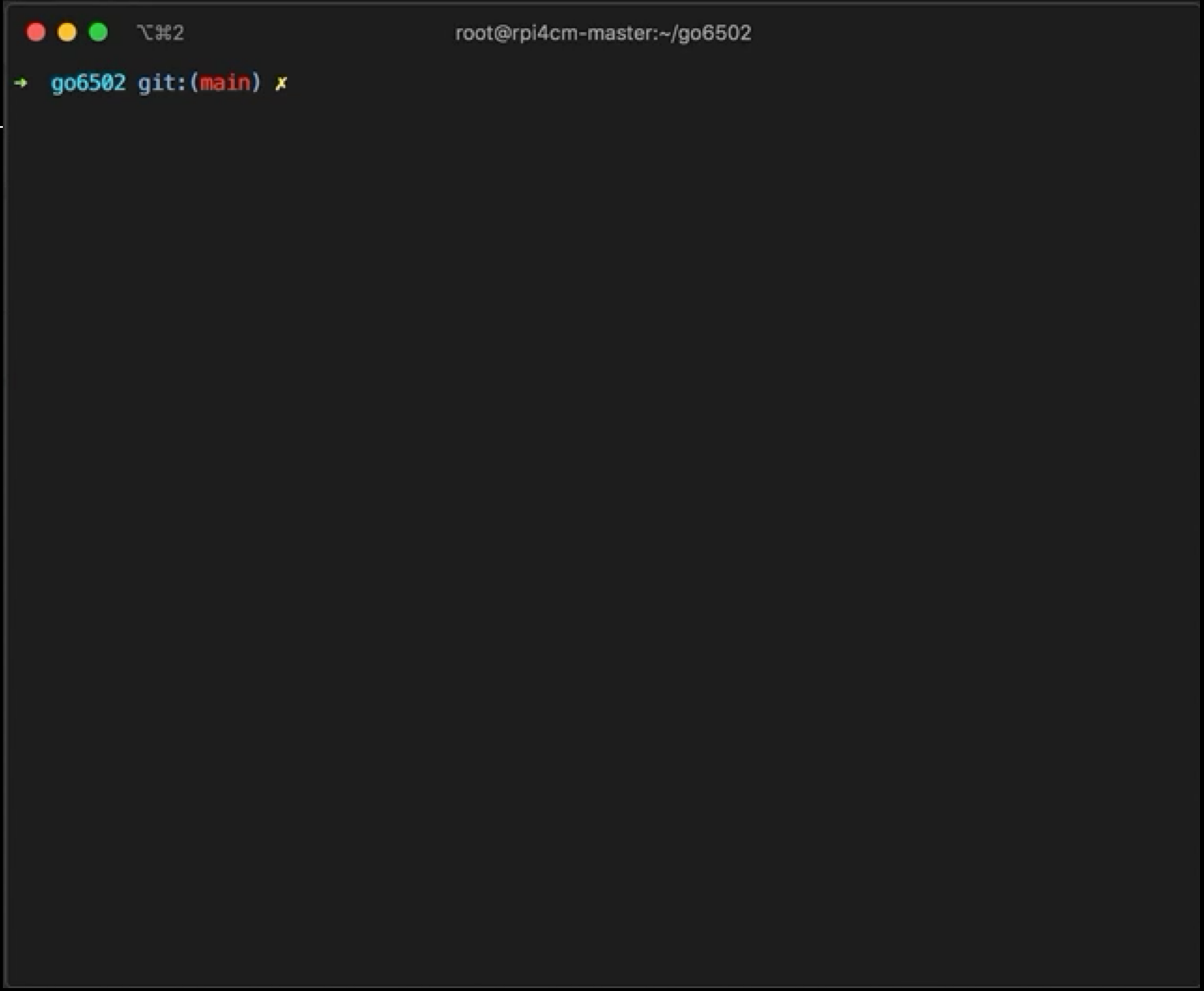



```
KBD = $C000
KBDSTRB = $C010
ASCBS   = $08 | %10000000

        ORG _TEXT
KEYIN   LDA   #$20
        LDY   CH
        STA   (BASL),Y    ;REPLACE FLASHING SCREEN
        LDA   KBD         ;GET KEYCODE
        CMP   #ASCBS      ;BACK SPACE? (CNTRL-H)
        BEQ   >1
        INC   CH
        INC   CH
^1      DEC   CH
        AND   #$7f
        STA   (BASL),Y
        BIT   KBDSTRB     ;CLR KEY STROBE
        RTS
```

Go模拟Apple II




```go
func (a *AppleII) WriteByte(pc uint16, b uint8) {

    if pc >= 0x400 && pc <= 0x7ff {
        // must be printable
        if cell := a.pixelMap[pc]; cell != nil {
            cell.SetText(string(b))
        }
    }

    if a.log != nil {
        a.log.Printf("WB 0x%04X -> %x", pc, b)
    }
    a.Mem.WriteByte(pc, b)
}
```



模拟Apple II 演示

Go模拟Apple II

- ZHUOS
  - Zhuo’ s Hardly Usable Operating System for fun
  - 几乎没法用的操作系统
- 小结
  - 早期电脑都是直接读写内存/硬件
  - 操作系统仅仅是帮助处理IO
    - 进程？不存在的
    - 用户？不存在的
    - 虚拟内存？不存在的
    - 都是用汇编编写程序

Go模拟Apple II

- ZHUOS
  - Zhuo’ s Hardly Usable Operating System for fun
- 小结
  - 早期电脑都是直接读写内存/硬件
  - 操作系统仅仅是帮助处理IO
    - 进程？不存在的
    - 用户？不存在的
    - 虚拟内存？不存在的

你说汇编？
可GCC 不支持6502啊

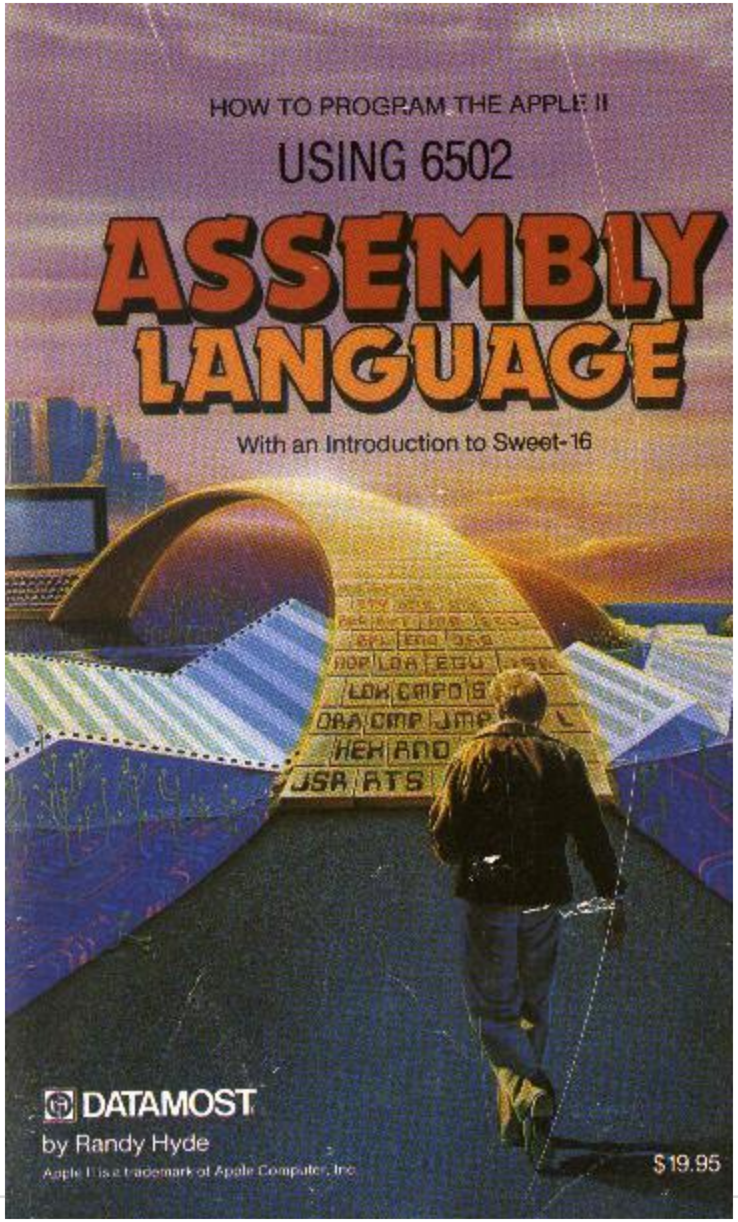
Go 编写汇编器

- 破除迷信
- Rob Pike：The assembler is just doing text processing

https://www.youtube.com/watch?v=KINIAgRpkDA

Go 编写汇编器

- 如何开始？
  - http://www.appleoldies.ca/anix/
- 语法参考Lisa（1982）
- 然后用Go实现！



Go 编写汇编器等工具

- 用Go编写的汇编器编写ZHUOS
  - Tokenizer
  - Parser（可以参加比赛了
  - 语法分析
  - 大概2000行左右就输出obj（json）
- 实现链接器
  - 最头疼的是指令地址确定
- 想debug方便还得
  - 实现objdump （检查指令和地址）
  - 实现nm（named map，检查链接）

```
KBD = $C000
KBDSTRB = $C010
ASCBS   = $08 | %10000000

        ORG  _TEXT
KEYIN   LDA   #$20
        LDY   CH
        STA   (BASL),Y    ;REPLACE FLASHING SCREEN
        LDA   KBD         ;GET KEYCODE
        CMP   #ASCBS      ;BACK SPACE? (CNTRL-H)
        BEQ   >1
        INC   CH
        INC   CH
^1      DEC   CH
        AND   #$7f
        STA   (BASL),Y
        BIT   KBDSTRB     ;CLR KEY STROBE
        RTS
```

Go 编写汇编器等工具

```bash
#!/bin/bash

set -ue

rm -rf zhuos/src/*.out

for f in zhuos/src/*.s
do
        if [ "$f" != 'zhuos/src/symbols.s' ]; then
            echo $f
            go run cmd/asm/main.go -i $f
        fi
done

go run cmd/link/main.go z.dat zhuos/src/*.out
```

⌥⌘2 root@rpi4cm-master:~/go6502

```
➜ go6502 git:(main) ✗ ./buildos.sh
```

编译ZHUOS演示

未来计划

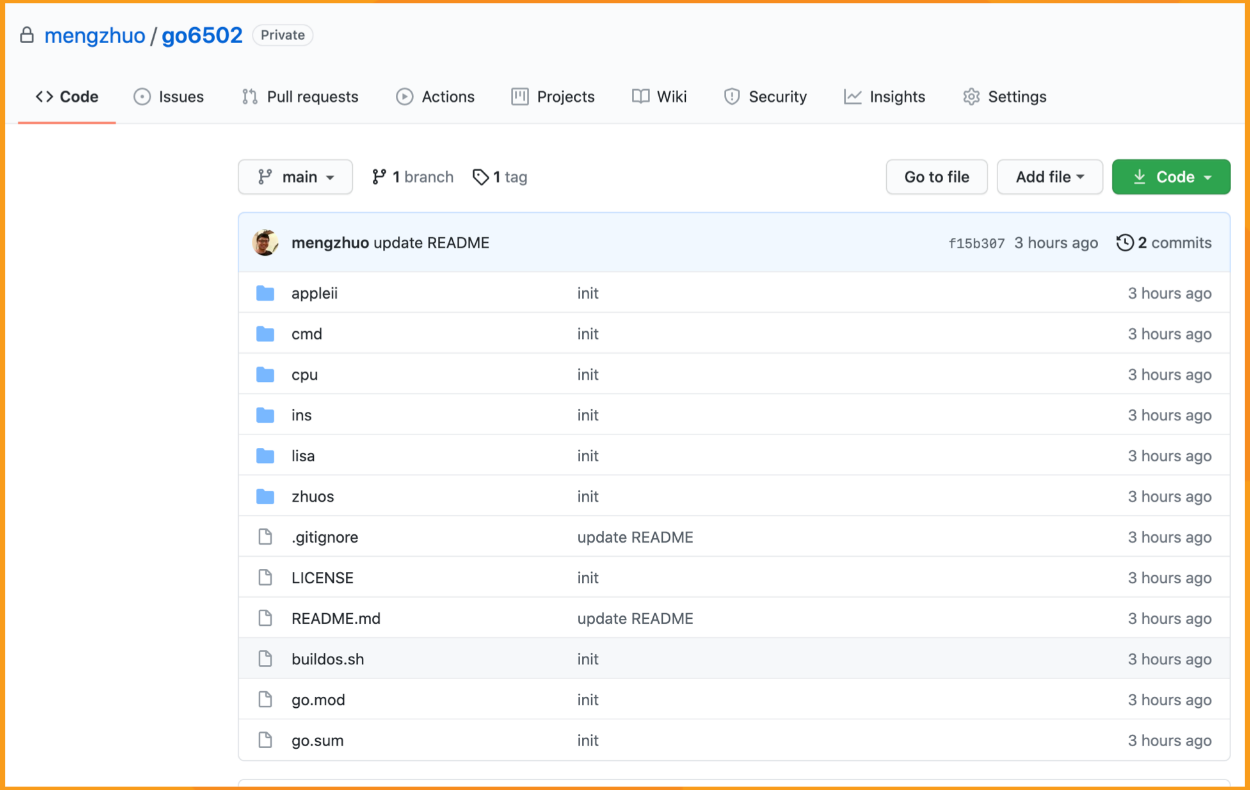
- 让Go能跑在6502上（10%）
  - 已经能输出部分语句
  - Go语法上除了直接操作指针之外跟C基本没区别
  - Runtime/map/atomic/channel/goroutine估计没戏
- The compiler is just doing text processing for lazy people

Thanks and Happy Hacking

- 图片除已声明或带链接的外均来源于Wikipedia
- 本次演讲不代表华为公司立场

One more thing

开源Go6502



https://github.com/mengzhuo/go6502