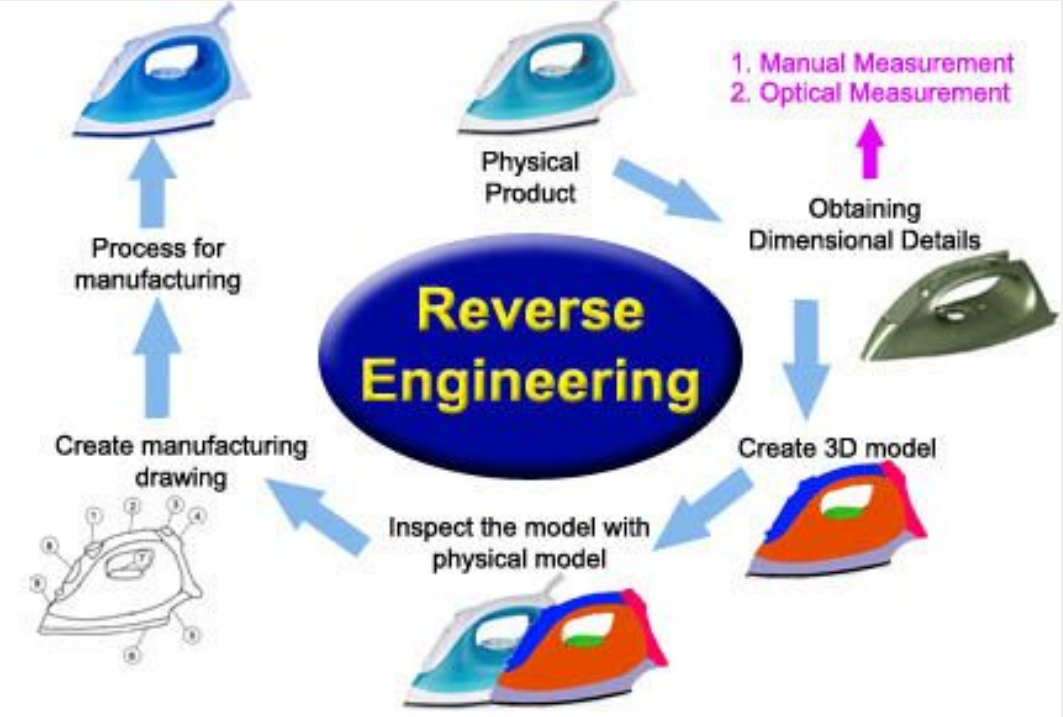
# Reverse engineering basics

@KirilsSolovjovs on twitter
http://kirils.org for more

Mg.sc.comp. Kirils Solovjovs
Possible Security

# Reverse engineering?



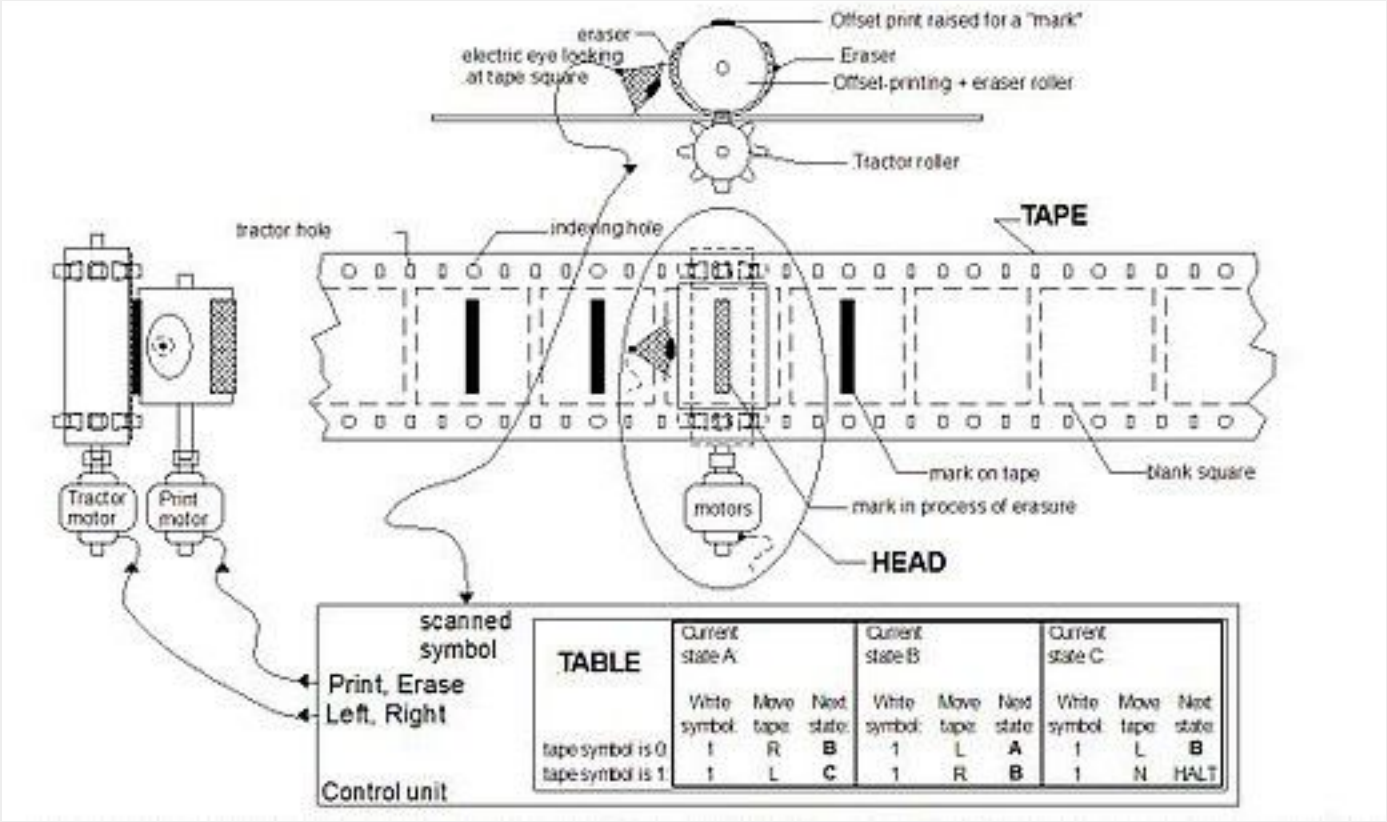

www.indiamart.com

# Contents

- Hardware architecture
- Processors and machine language
- Engineering: Creating a program binary
- Reverse engeineering:
  - Static analysis
  - Binary debugging

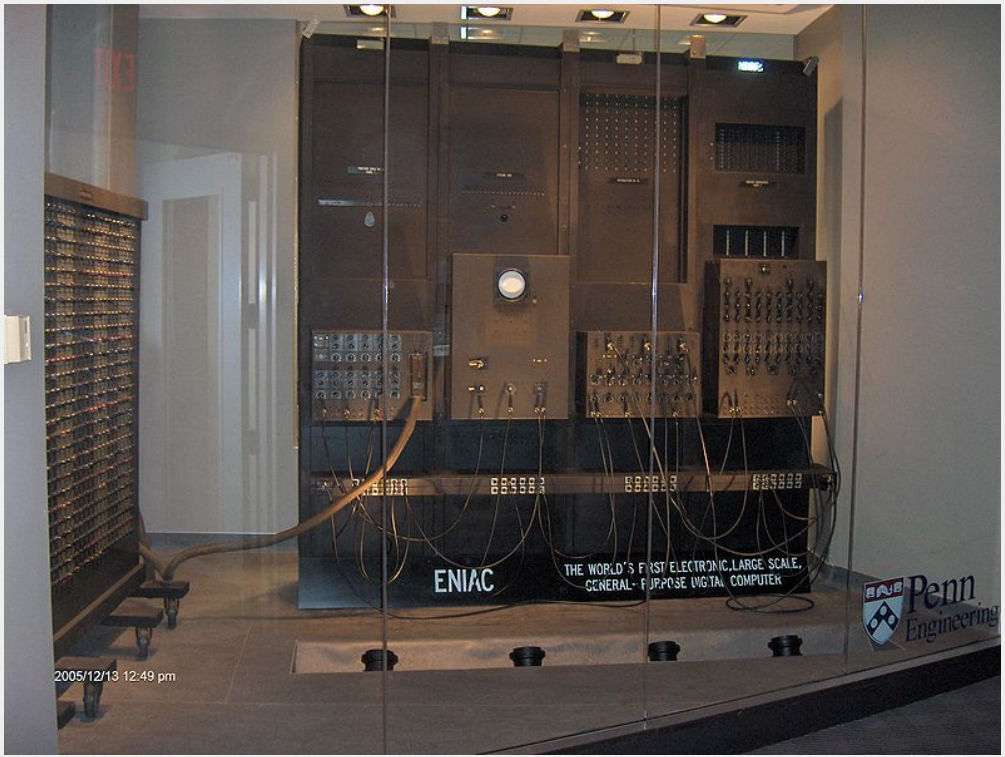
# Hardware architecture

# Theory. Turing machine



# ENIAC, 1945

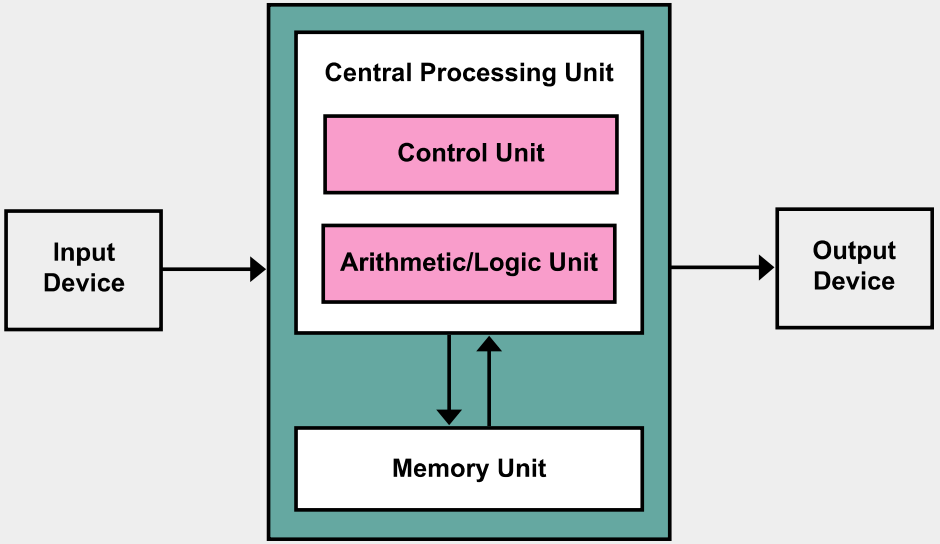
- Turing complete
- General purpose
- “Reprogrammable”
  - Physical rewiring required
  - Takes weeks



# The two common hardware architectures

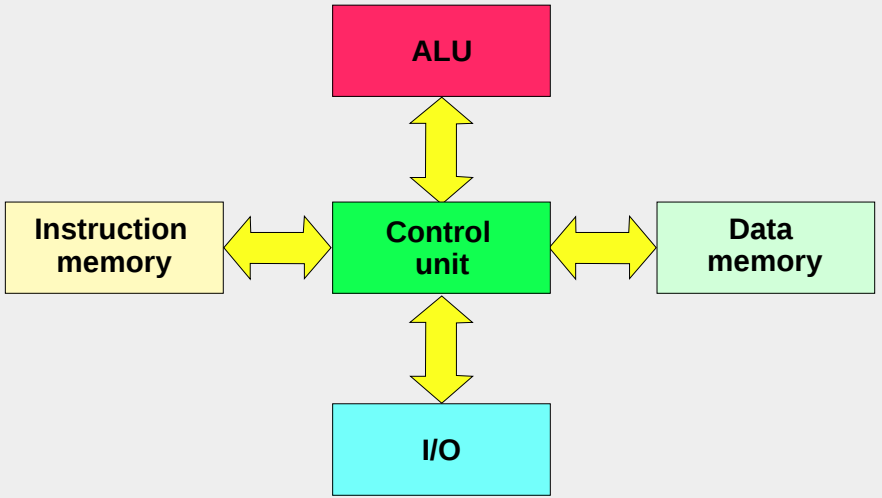
# Von Neumann arch

- “Stored program” concept
- CPU = CU + ALU
- Joint MU
- I/O
- Most modern systems



# Harvard arch

- Separate CU and ALU
- Separate memories
  - Data
  - Instructions
- Allows memories to have different attributes
- Improved speed
- DSPs, some microcontrollers



# Machine code

- A list of machine language instructions to be directly executed by a CPU.
- Each instruction is a small specific task:
  - Data operations
  - Arithmetic and logic operations
  - Control flow operations
- Different ISAs (Instruction set architectures):
  - 8086, ARM, MIPS, VAX, ...

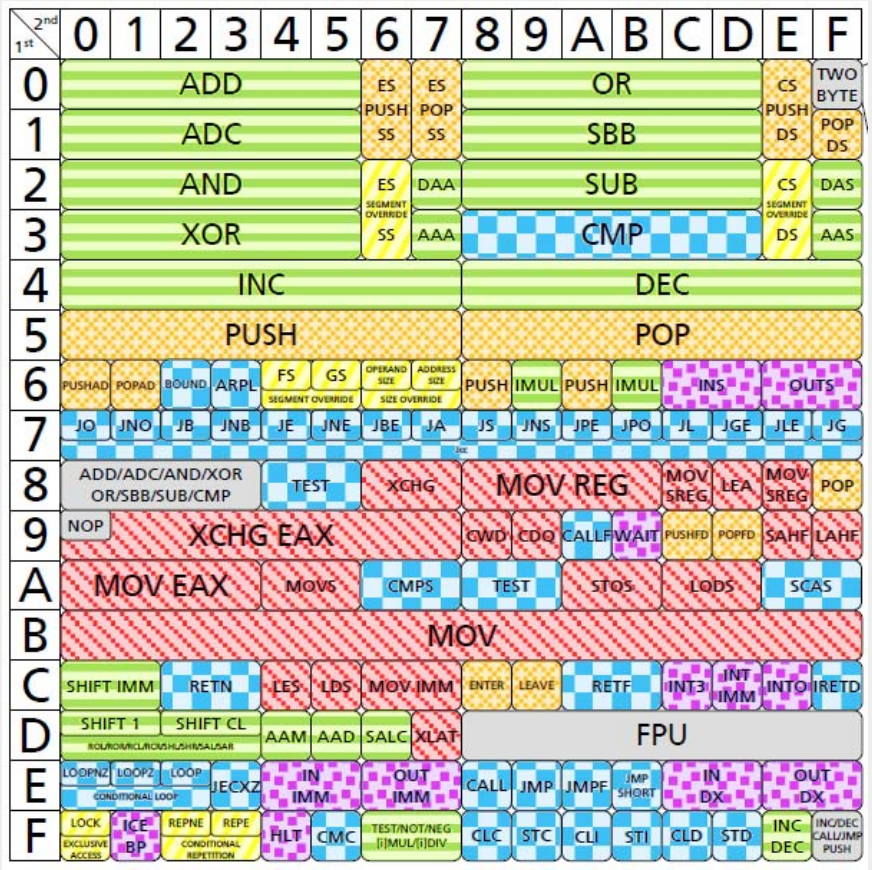
# Instruction

## MIPS32 Add Immediate Instruction

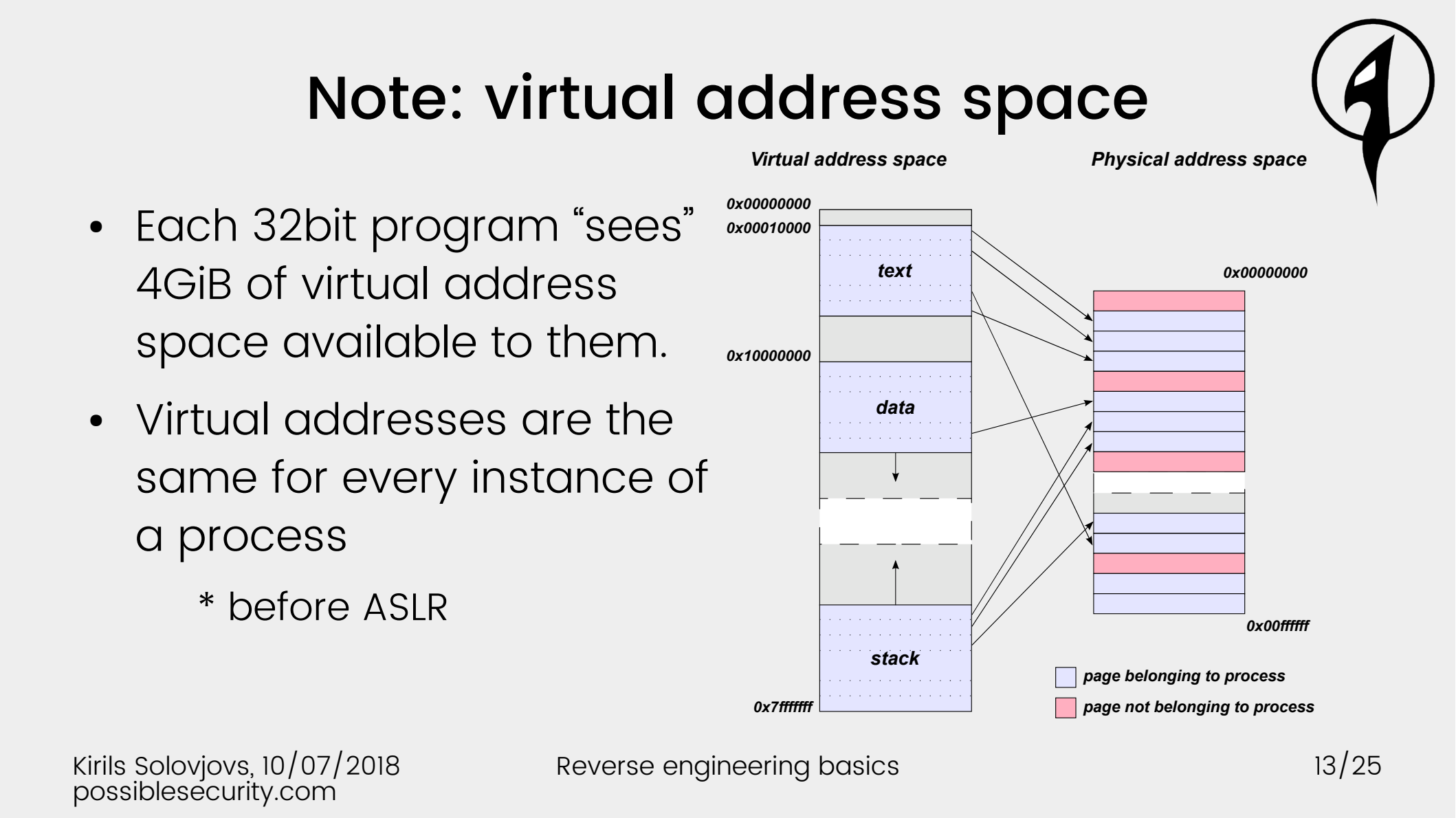
| 001000 | 00001 | 00010 | 0000000101011110 |
|---|---|---|---|
| OP Code | Addr 1 | Addr 2 | Immediate value |

Equivalent mnemonic: **addi $r1, $r2, 350**

# x86 opcodes


1st \ 2nd
0 1 2 3 4 5 6 7 8 9 A B C D E F
0 ADD ES PUSH SS ES POP SS OR CS PUSH DS TWO BYTE
1 ADC SBB POP DS
2 AND ES SEGMENT OVERRIDE DAA SUB CS SEGMENT OVERRIDE DAS
3 XOR SS AAA CMP DS AAS
4 INC DEC
5 PUSH POP
6 PUSHAD POPAD BOUND ARPL FS GS SEGMENT OVERRIDE OPERAND SIZE ADDRESS SIZE SIZE OVERRIDE PUSH IMUL PUSH IMUL INS OUTS
7 JO JNO JB JNB JE JNE JBE JA JS JNS JPE JPO JL JGE JLE JG
8 ADD/ADC/AND/XOR OR/SBB/SUB/CMP TEST XCHG MOV REG MOV SREG LEA MOV SREG POP
9 NOP XCHG EAX CWD CDQ CALLF WAIT PUSHFD POPFD SAHF LAHF
A MOV EAX MOVS CMPS TEST STOS LODS SCAS
B MOV
C SHIFT IMM RETN LES LDS MOV IMM ENTER LEAVE RETF INT3 INT IMM INTO IRETD
D SHIFT 1 SHIFT CL ROL/ROR/RCL/RCR/SHL/SHR/SAL/SAR AAM AAD SALC XLAT FPU
E LOOPNZ LOOPZ LOOP CONDITIONAL LOOP JECXZ IN IMM OUT IMM CALL JMP JMPF JMP SHORT IN DX OUT DX
F LOCK EXCLUSIVE ACCESS ICE BP REPNE REPE CONDITIONAL REPETITION HLT CMC TEST/NOT/NEG i]MUL/i]DIV CLC STC CLI STI CLD STD INC DEC INC/DEC CALL/JMP PUSH

# Note: virtual address space

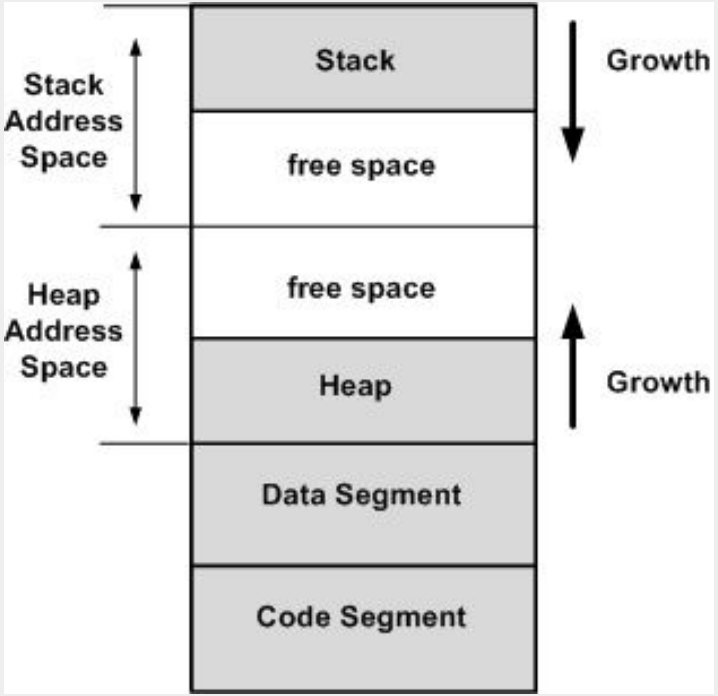


- Each 32bit program “sees” 4GiB of virtual address space available to them.
- Virtual addresses are the same for every instance of a process

    * before ASLR

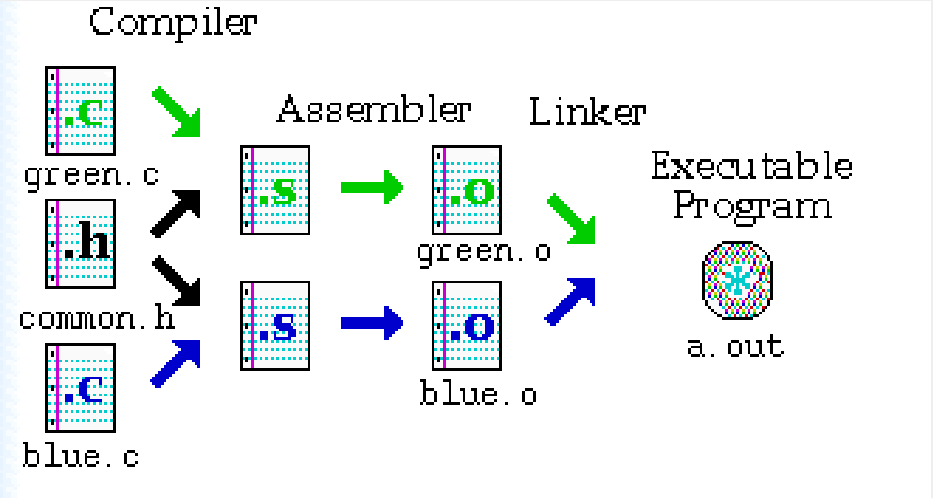
# Stack and heap

- Where are variables stored then?
  - Stack and heap
- Heap: for dynamic allocation, random access
- Stack: for static memory allocation



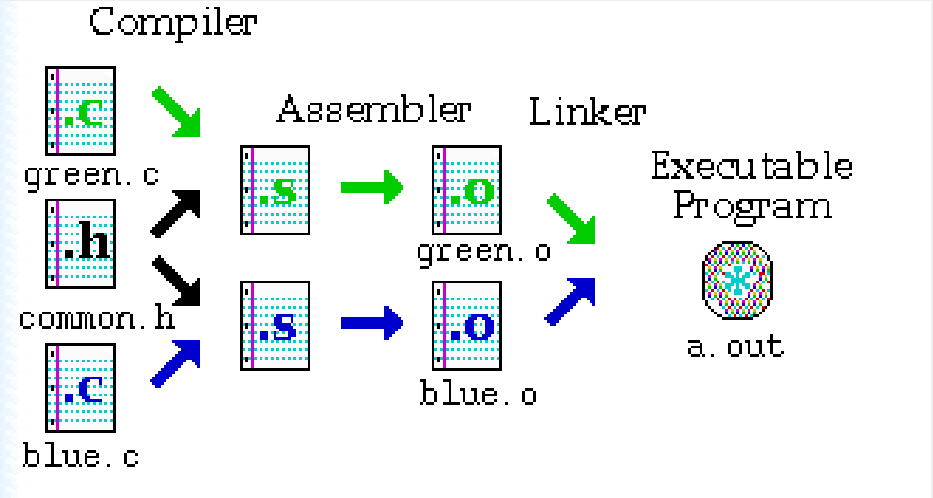
# Creating a program binary

# Overview: gcc example




- .c, .h – human readable C
- .s – human readable assembly
- .o – binary object code (relocatable object code)
- a.out – directly executable machine code

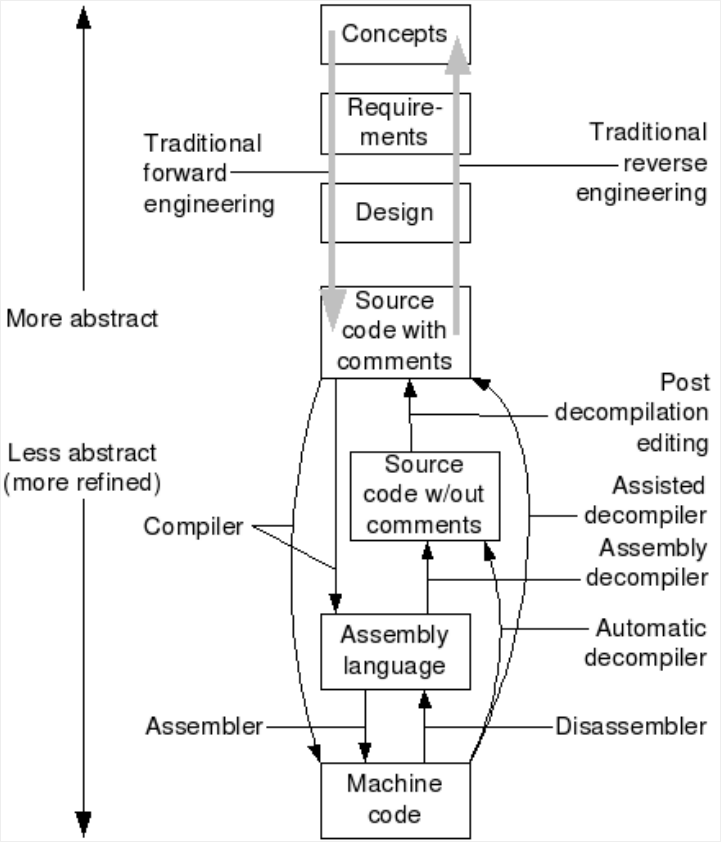
# DEMO: gcc example




- Compile:
  - gcc -S file.c -o file.s
- Assemble (&optimize):
  - gcc -c file.s -o file.o
- Link:
  - gcc file.o -o file

# Reverse engineering

# Full cycle


Concepts
Require-
ments
Design
Source
code with
comments
Source
code w/out
comments
Assembly
language
Machine
code
Traditional
forward
engineering
Traditional
reverse
engineering
More abstract
Less abstract
(more refined)
Post
decompilation
editing
Assisted
decompiler
Compiler
Assembly
decompiler
Automatic
decompiler
Assembler
Disassembler

# DEMO: Static analysis of password

- strings r2
- r2 password
  - pdf @ main

# DEMO: Static analysis of Android APK

- binwalk
- dex2jar + jd-gui

# DEMO: Binary debugging of test42

- gdb test42
  - info files
  - b *main
    - if not stripped
  - start
  - info registers
  - x/i $pc

# DEMO: Firmware reverse engineering

- Real life example – mt

# Overview of tools

- gdb
- Capstone
- radare2
- IDA-Pro & Hex-Rays
- Binary Ninja
- OllyDbg
- otool
- PE Explorer
- binwalk
- dex2jar & JD-GUI
- Resource Hacker

# Thanks!

Slides are available on http://kirils.org

Find me on twitter: @KirilsSolovjovs