NachOS

An Introduction to System Programming

Uni Karlsruhe
Systems Chair / IBDS
Christian Ceelen

Administration

- subscribe mailinglist at
  - http://lists.ira.uka.de/mailman/listinfo/nachos
  - Send mails to: nachos@ira.uka.de
- form groups with 2-3 persons
- registration will close with the 2nd assignment
  - announcements in the lecture and mailinglist

I hear and I forget,

I see and I remember,

I do and I understand.

-Chinese Proverb

What is the lecture about?

- Theoretically
  - design a complex system
  - introduce abstractions for system components
  - show different possible designs
  - compare different design decisions
  - research which architecture suits for different environments
  - research tradeoff effects
  - review common implementation issues/bugs

I hear and I forget,

I see and I remember,

I do and I understand.

-Chinese Proverb

Programming Assignments

- Provide introduction into „System Programming“
- Provide deeper understanding of Operating Systems through practical experience.

- Approach: Participate in the implementation of a simple operating system.

Goal

- Programming with C/C++ (and [illegible])
  - see paper at our home[illegible]
  - deeper knowl[illegible] gcc/g++/gasm/ld not necessary
- Use diff[illegible] programming tools
  - [illegible]per knowledge on make not necessary
  - try to use a versioning system
- Work in a team
- Learn how to debug a bigger project
- Learn how to debug foreign code
- Learn how to debug your own code

You should try to work this out!

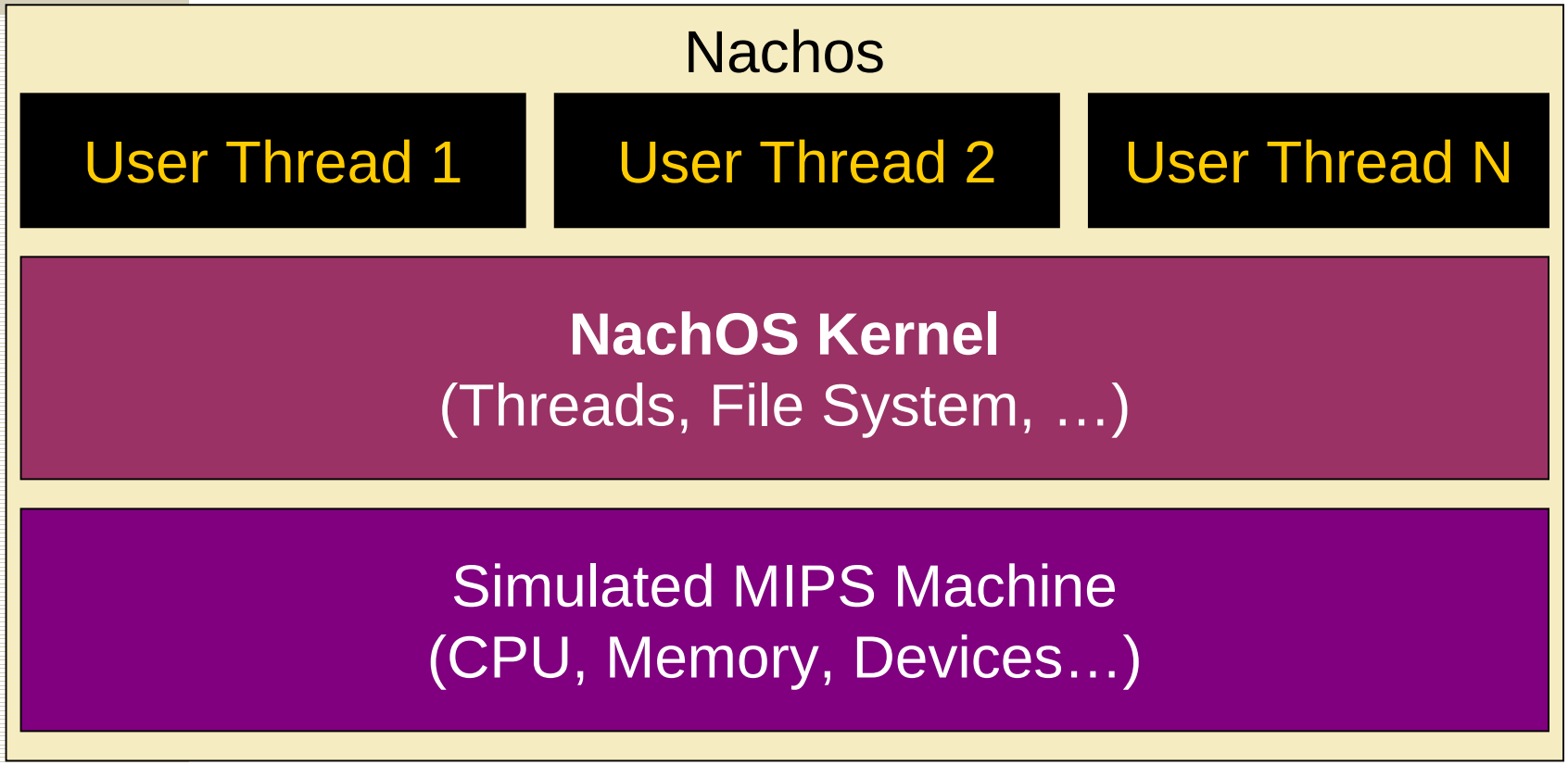
NachOS

- NachOS:
  - Not Another Completely Heuristic Operating System
- Written by Tom Anderson and his students at UC Berkeley
  - http://www.cs.washington.edu/homes/tom/nachos/

NachOS

- An educational OS used to
  - teach monolithic kernel design and implementation
  - do experiments
- Fact:
  - Real hardware is difficult to handle.
  - May break if handled wrong.
- Approach:
  - Use a virtual MIPS machine
  - Provide some basic OS elements

NachOS V4.0



Base Operating System

NachOS V. 4.0

- Simulates MIPS architecture on host system (Unix / Windows / MacOS X ?)
  - User programs need a cross-compiler (target MIPS)
- Runs multiple NachOS threads as one Unix process
- Nachos appears as a single threaded process to the host operating system

Exercises

- Ex1 – Thread Synchronization
- Ex2 – System Call
- Ex3 – Virtual Memory Management
- Ex4 – File System

Challenge:

- Ex5 – Optimization

What‘s in for you?

- Just participate:
  - Good training for the exam
  - Fun
- Complete assignments 1-4
  - bonus points for the final exam
- Win the challenge (assignment 5) and get
  - Free tickets for a concert of “Peter and the Wolves”
  - Meal at the Vogelbräu (including beer!)

What to do?

- Assignments 1-4
  - due in the last week of this term
  - short code review
- Assignment 5
  - due in the last week before the exam
  - short code review
  - better performance than every other group

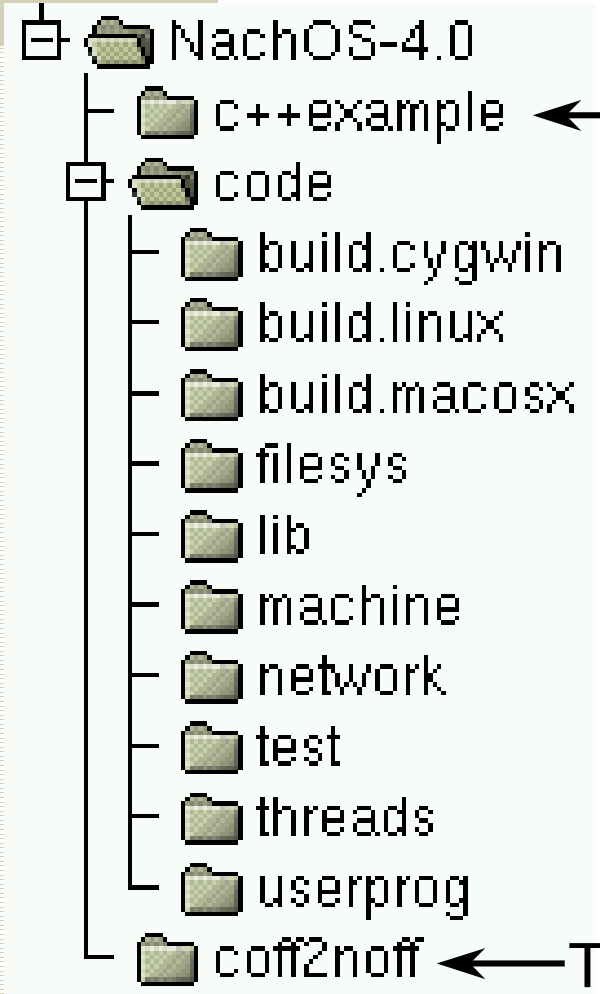
Prerequesits

- Linux
  - x86
  - host gcc
  - Make
  - cross compiler for decstation-mips
- Windows
  - x86
  - Cygwin (→ http://www.cygwin.com)
  - cross compiler for decstation-mips

Setup your System

- Get tar-ball from our homepage
  - Get NachOS-4.0.tgz
  - Get Cross-Compiler (mips-x86.*-xgcc.tgz)
    - Install as root (/usr/local/nachos....)
  - Or build a cross compiler
    - Get binutils and gcc source from gnu
    - „configure --target=decstation-mips --prefix=$Install-path“
    - „make“ and „make install“ (first binutils, then gcc)

NachOS content



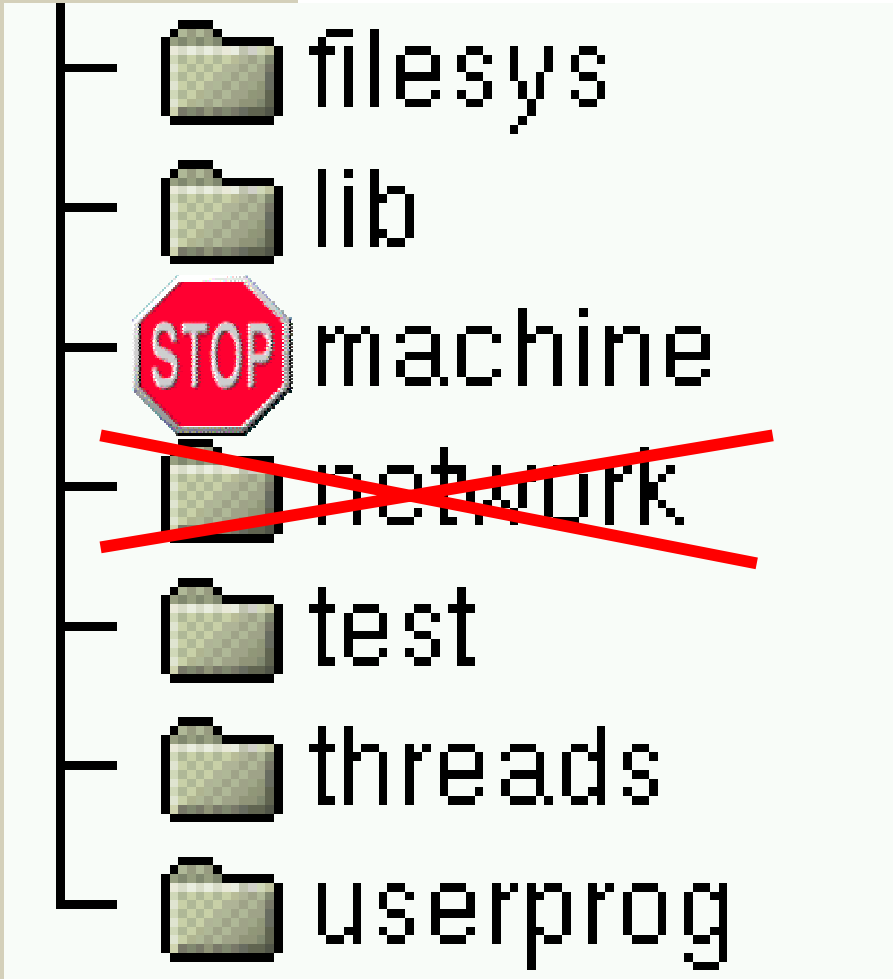
C++ introduction

Build directories:
make depend
make

Actual NachOS source of the kernel and the simulator

Tool needed to convert user programs

NachOS Code



~ Assignment 4

Some helper functions.

Simulator! Do not change!

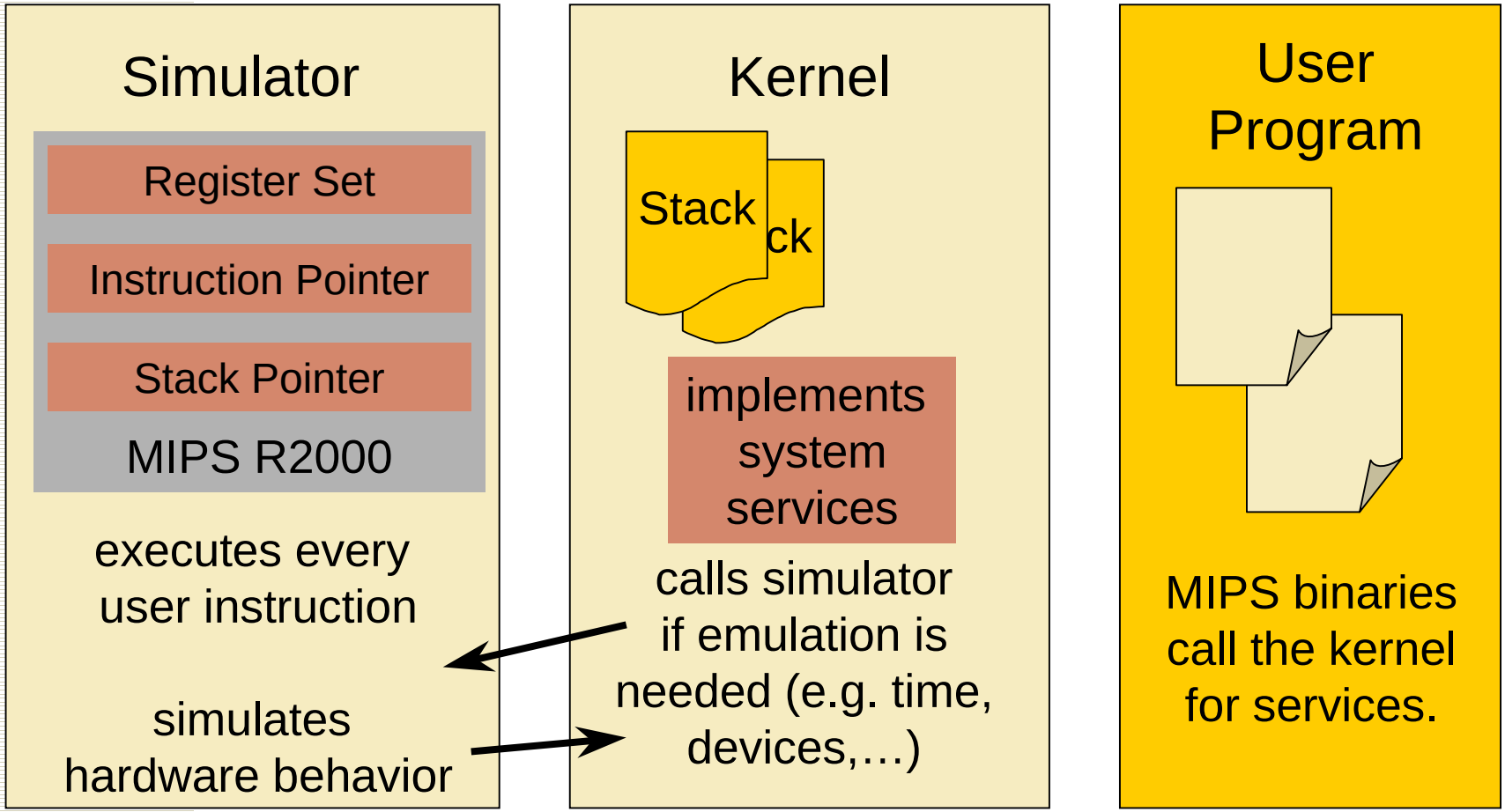
Don‘t bother about this.

NachOS test applications

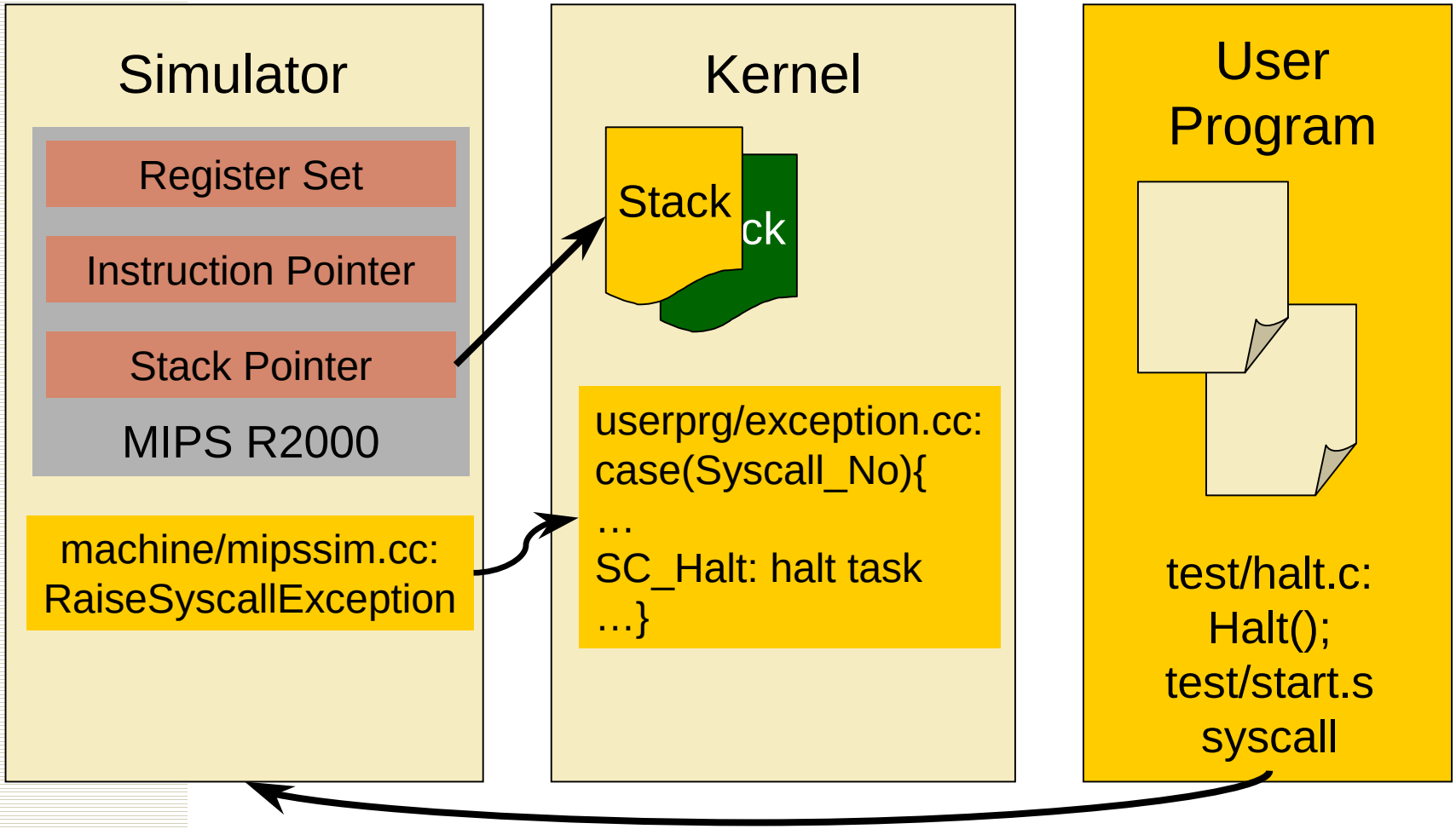
Assignment 1

~Assignment 2 & 3

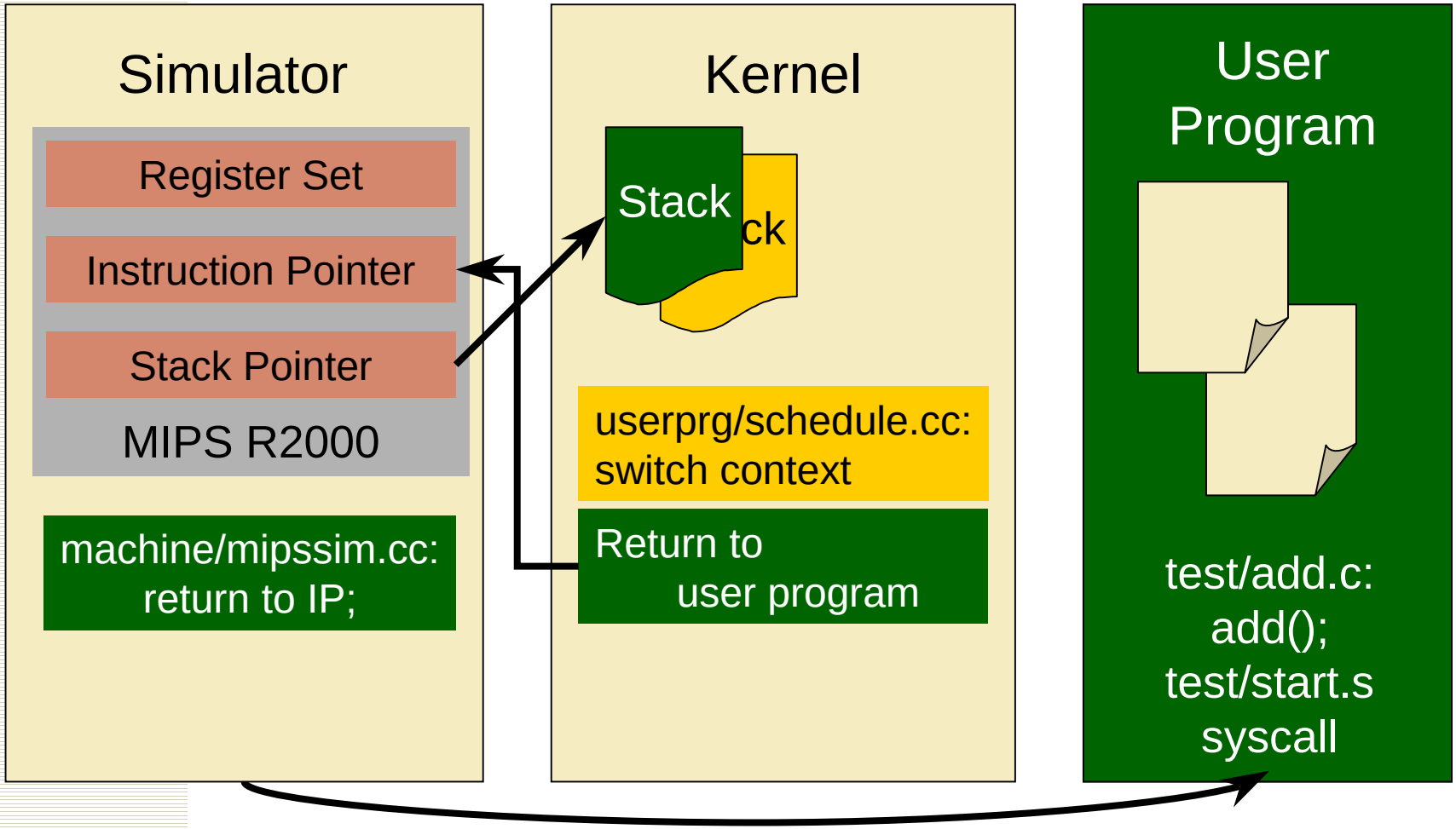
How does it work?



How does it work?



How does it work?



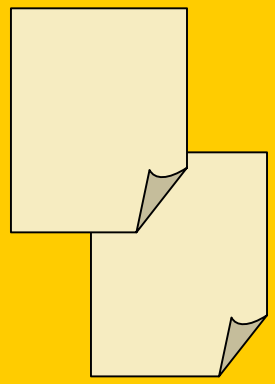
How does it work?

Simulator

Kernel

- Runs natively on host OS. Needs gcc & binutils (included in Cygwin or Linux)
- Basic OS services are already implemented
- Simulates hardware behavior (don't need raw device access!)

User Program



MIPS binaries
Needs
cross compiler
and NOFF-tool

test/start.s

- MIPS dependend assembler code for userlevel bindings
- **Do not modify!**
- How does it work:
  - gets parameter via registers (C-calling convention)
  - loads syscall number into first register
  - does a syscall exception (enter the kernel)
  - jump back

threads/switch.s

- Host Machine dependent assembler code for context switches
- **Do not modify!**
- How to perform a context switch:
  - SWITCH(oldThread, newThread);
- How does it work:
  - Saves current “register set” to stack
  - Changes stack pointer
  - Loads register set from new stack and returns

Coding Philosophy

- Write everything at once, then start d g
  - Big Bang
  - Write test code and doc
- Write everything i al and check each step
  - D ry step
  - est code while programming each step

Both approaches work!
But which does work for you?

Your System is unstable.
Internal Windows Error #231
Please Reboot your machine

System Programming

- Document your code!
- Write reentrant functions! (no global variables)
- Optimization:
  - Don‘t do it on the first run.
  - Get the design and abstraction right in the first place!
  - A good design yields a better performance with few optimizations.
- Do you know what your tools are doing?

Debugging

- Included debugger supports:
  - Single stepping (option –s)
  - Tracing (option –d td prints debugging messages of thread „t“ and disk „d“ emulation)
- Using gdb with NachOS
  - needs compiler option (–ggdb)
  - http://www.student.math.uwaterloo.ca/~cs354/misc/debug.html

Questions ?

Think first,
code later!

Follow the instructions given to you!
They should help you!