Concepts

**Abstraction**: with multiple levels of detail / description, it helps manage complexity and provides interfaces and standards.

**Virtualization**: gives the appearance of a capability or service. This decouples the service from the underlying physical resources. It helps in terms of simplicity, flexibility and resource sharing.

4 bits – nibble, 8bits – Byte, 2 Bytes – Word,

**Endianness**: the order the bytes within a single word are stored within main memory

Big endiannes -- most significant byte first

Little .. -- least ..

**prototypes**: signatures of functions will be used in code

**static** **keyword** reduces the scope to the file it is within

Number System

Hex: Aa-10 Bb-11 Cc-12 Dd-13 Ee-14 Ff-15

SI decimal prefixes -- kilobyte 103

IEC binary prefixes -- kibibyte 210

**Mbps**: mega bits per second

**MIPS**: million instructions per second

**FLOPS**: floating-point operations per second

**2-complement**: flip all bits , add one

**Convert fractional number** to fractional binary number --

keep times 2 and get the integer bit as the binary bit.

**Floating point numbers**

(-1)smbe, s sign bit; m significand; b base; e exponent

IEEE 754. **single precision numbers**:

1 bit sign/ 8 bits exponent/ 23 bits significand

**other three types**:

- **subnormal numbers**, exponent is 0x00: (-1)s\*0.m\*2(-126)

- **normalized** **numbers**, exponent between 0x01 and 0xFE:

(-1)s\*1.m\*2(e-127)

- **special** **numbers**, exponent 0xFE:

if m = 0, +- infinity; else NaN

**%**, the way it works on negative numbers is not exactly specified in c

**'=='** on floating point numbers indicate problems

x^y xor

**Arithmetic** right shifts fill the left most bits with sign bit

**Logic** right shift fill ... with 0.

LINUX

**ltrace'** lists the library calls made by a running process as it is running. **'strace'** list the system calls. A file is a named collection of related information.

**touch**: change timestamp / create an empty file

Files

A file is a named collection of related information.

A file system provides a uniform logical view of related information.

Two ways to read / write files:

- system calls; open, read, write, ioctl

- file streams, fopen, fread, fwrite

A file can be mapped into memory, this can be done with the 'mmap' function. 'mmap' enables the file's contents to be view and modified in normal memory.

SHARED mmap: modifications to the memory of the mapped file are written back to the actual file, also other processes that maps the same file sees the same modifications

PRIVATE mmap: process has its own private copy of the file. Modifications are not written back to the file and other processes do not see any changes

fixed file descriptor numbers: stdin is 0,stdout is 1, stderr is 2

res = read(0,buf,255); res = write(1,"Hello",6);

methods to access files:

sequential access

direct access

memory mapped access

**open**:

returns a 'file descriptor' which is a small non-negative number which provides a reference to the opened file

int open(const char \* pathname, int flags, mode\_t mode)

The OS keeps a table of the files that each process has opened.

**read**:

ssize\_t read(int fd, void \* buf, size\_t count)

returns 0 at the end of file

**truncate**, **ftruncate** resize a file

int truncate(const char \* path, off\_t length)

int ftruncate(int fd, off\_t length)

**lseek**:

move current file offset

off\_t lseek(int fd, off\_t offset, int whence)

**ioctl**:

<sys/ioctl.h>

int ioctl(int d, int request, ...)

control opened devices

**stat**:

provides file information

<sys/types.h>

<sys/stat.h>

<unistd.h>

int stat(const char \* path, struct stat \* buf)

int fstat(int fd, struct stat \* buf)

int lstat(const char \* path, struct stat \* buf)

**getchar** returns EOF integer when reaches the end of the file

**scanf** returns the number of items successfully matched; or EOF at the end of file

Structure of a c program

Source 🡪(COMPILE)🡪 assemble code(.s)🡪 (ASSEMBLE)🡪object code(.o)🡪(LINK)🡪executable

**malloc** returns a pointer to the amount of memory requested; need free

**struct** may add padding to make certain types are appropriately aligned in memory

rPeAnut

xvar : block #1 ; apply for a global variable xvar initialized to 1

xvar: block 10 ; a global array xvar of length 10, all values 0

str: block #”abcde” apply for a string

load #xvar R0 ; load address of xvar to R0 (immediate load)

load xvar R0 ; load the value in xvar to R0 (absolute load)

store R3 xvar ; assign value to global variable xvar (indirect restore)

store R3 #xvar R0; global variable as base

store R0 #-1 SP ; store for return value in recursion

p1.x = 4 differ to pp1 -> x = 4 in what you load from

p1.x load from #p1 , p1 -> x load from pp1

**negate** -> flip all bits then plus 1, **not** -> just filp all bits

rotate: most significant bit will turn back to the least bit?

status 0xFFF1, IO control 0xFFF2 represents?

memory that is addressable in words of 32 bits

total maximum amount of addressable memory

216 = 65523 words or 262144 bytes

(although only 3 are actually used: interrupt vectors)

once microprocessor reset, IP is set to 0x0100

MAR(memory address register); MDR(memory data register); bus control signals

**To read from main memory**:

- CPU places address it wishes to read from into MAR

- MAR is placed on the address bus, a read signal is given to the bus

- memory reads the address and writes to data bus

- contents of the data is coped into MAR

**To write to memory**:

- CPU places address it wishes to write to into the MAR, also data is placed into MDR

- MAR, MDR is placed on bus and write signal is given to bus

- memory write the data to the address

CPU contains

- control unit

- 8 general purpose registers

- 3 constant registers: ONE, ZERO, MONE

- IR, holds the current instruction is being executed

- status register(SR), status information about CPU

bit 0: overflow (OF)

bit 1: interrupt mask(IM)

bit 2: enable timer interrupt(TI)

**16 bit registers:**

- stack pointer (SP) pints to top of stack used for method calls, returns interrupts

- program counter(PC), address of next instruction

- arithmetic logic unit (ALU)

MAR, MDR, IR, PC are not directly accessible via instruction set

**instruction set:**

are all 1 word long (32 bits)

- address and values take 16 bits of the 32 bits

- value are sign extended from 16 bits to 32

take care with bits 15 to 31, **load immediate** will not directly work

To read from terminal, first check a character is available to read from

readc: load 0xfff1 R0

jumpz R0 readc

load 0xfff0 R1

**Stack frame:**

during execution of a function, the executing code will need to be able to access:

return values (sometimes)

parameters (sometimes)

return address

local variables (sometimes)

calling code will create part of the new stack frame, once the procedure completes the part of the stack frame, the calling code created is removed via the calling code

**stack frame in rPeANUt:**

return value (if there is one) - added by calling code

parameters - added by calling code

return address - added using the 'call' instruction

local variables - added by procedure code

**calling code** will:

- push to add a spot for return value

- push to add parameters for teh call

- call command

- remove added parameters (pop)

- remove return value (pop)

**procedure code:**

- push to add requires local variables

- execute

- remove local variables (pop)

- return

Interrupts

Interrupt, an event alters the normal fetch-decode-execution. When occurred, current program will be suspended. caused by hardware event and software event.

Interrupt handler should be carefully designed such that the state of CPU returns exactly how it was found.

Interrupt latency, time between an interrupt occur and code executed to handle it.

Interrupts occur more frequently than the handler can serve, then requests to handle interrupts will be lost (interrupt storm)

Functionality of Reset IM?

Memory

Dynamic random access memory (DRAM) only requires a single transistor and capacitor per bit, this enables much higher density memory (compared with SRAM). However, it is considerably slower and more complex to access.

Static random access memory (SRAM) is fast volatile memory that is simple to interface with, however, it uses a lot more area than DRAM.SRAM will be found in: CPU caches, buffers for various devices within a computer system.

ROM is non-volatile so state is maintained without power.

**Flash memory** is an electronic [non-volatile](http://en.wikipedia.org/wiki/Non-volatile_memory) [computer storage](http://en.wikipedia.org/wiki/Computer_storage) device that can be electrically erased and reprogrammed.

registers

L1 cache capacity increase ↓

L2 cache speed↑

DRAM

Flash Drive

Magnetic Disk Drive

Network Backup Server

Building a CPU

two basic approaches for the design of the control unit

Hardwire - these control units basically are finite state machines that are specially designed to sequence the CPU based on the instruction it is executing. (not particularly flexible and can be difficult to design, however, good performance)

Microcode - this approach uses ROM type memory within the CPU to act like machine code for sequencing the different action within the CPU. simpler and more flexible approach.

Performance

PUs have moved caches onto the CPU die which enables the CPU to be physically closer to the cache. This reduces latency.

Pipeling: involves in a number of independent stages, IF, ID, E, WB

Superscale architectures involve duplicating functional units within the cpu and then starting more than one instruction on the same clock cycle in the pipeline. This enables a larger throughput of instructions.

Sometimes instructions will require data from memory before they can execute, this will stall the pipeline. Slow the CPU.

The "Outer of order execution" approach loads the next few instructions and starts executing the instruction that has the required data

Multithread: CPUs can maintain the programming context of multiple threads (so duplication of state and register information), without the duplication of processing units, caches, TLBs, etc, This enables multiple threads to be executed within the one core. This can hide latency.

Multicores: The CPU can be duplicated, share the same memory and L2 Cache.

.

Linking

1. This binding of instructions and data to memory addresses may occur at :

compile time, load time, or execution time.Li

2. Libraries may be either:

statically linked - the library is compiled into the final binary executable.

dynamically linked - just a 'stub' is includes in the binary executable, the library code is obtained as needed during execution.

processes

A **context switch** changes the process a CPU is executing.

A **thread** is the basic unit of execution within a process.

Each thread, lightweight process(**LWP**), consists of a program counter, a register set and a stack space

A traditional process is called **heavyweight** process

= task with one thread

A group of threads combine together to form a task/ process. they **share** text, data, opened files, capabilities, etc.

A **CPU-cache** stores copies of main memory data.

cache (SRAM), memory (DRAM)

**working** **set** is the memory that a program is currently using

fully associative cache

direct mapping cache

set associative cache

Memory Management and Paging

Addresses generated by the CPU are referred to as logical addresses. These are the addresses 'seen' by the user's programs.

Addresses seen by the main memory are referred to as physical addresses.

Physical memory is partitioned into fixed-sized blocks called frames.

Logical memory is partitioned into blocks of the same size called pages.

Logical addresses (produced by the CPU)are divided into two parts: the page number and the offset.

Each process has a page table.

A user or process would like to view memory as a set of variable-sized segments. Each segment has a name and there is no necessary ordering among segments.

Advantages of separating logical & physical addresses:

- execution-time address binding

- simplifies the swapping of processes in and out of memory

- protection / security of data between processes

- simplifies sharing of data

Virtual Memory

Swapping is an approach that involves bringing an entire process from disk into main memory so it may be executed.

Demand-paging is a lazy "swapper" which brings into memory the pages of the process that are needed.

One approach to paging is to only bring pages into memory when they are needed. This is called Pure demand paging.

advantages:

- programs can be larger than physical memory

- abstracts main memory into an extremely large logical storage area

- increases the degree of multi-programming

valid-invalid bit: hardware support

**swap** **space** generally is faster than the file system( as file lookups and indirect allocation methods are not used)

page replacement methods:

- frame-allocation algorithm

- page-replacement algorithm(FIFO, OPT/MIN, LRU)

CPU Scheduling

In [*multiprogramming*](http://en.wikipedia.org/wiki/Multiprogramming) systems, the running task keeps running until it performs an operation that requires waiting for an external event (e.g. reading from a tape) or until the computer's scheduler forcibly swaps the running task out of the CPU. (multiprogramming is a CPU scheduling strategy.)

There are two types of schedulers:

non-preemptive - The CPU is not 'forcefully' taken from the process, and a

preemptive - The CPU may be forcefully taken from the process and switched to another process.

A variety of criteria are used to compare different scheduling algorithms.

CPU utilization - The percentage of time the CPU is in use.

Throughput - The rate at which processes are being completed.

Turnaround time - The interval of time from the starting a process to completing the process.

Wait time - The amount of time a process spends in the ready queue.

Response time - The amount of time it takes a process to start responding. This does not include the time to output the

response.

Three algorithms for CPU Scheduling:

First Come First Serve (FCFS)

Shortest Job First (SJF) provably shortest waiting time

Round Robin (RR)

Networks

Most Ethernet controllers have unique 48bit MAC address.

Ethernet's topology started as bus topology. But now LAN would generally use star topology with a switch at the center.

Weakness of bus topology: Only one device may transmit information to the bus at any time. If two units transmit at the same time a collision will occur, both transmission with usually be compromised and will need to be re-transmitted.

UDP/TCP

0.0.0.0 - This host (only source).

127.X.X.X - loopback.

10.X.X.X - private network

192.168.X.X - private network

255.255.255.255 - Broadcast on the local network.

IPv4 (Internet Protocol) provides 32bit addresses (decimal) for addressing computing devices that span the globe.

TCP(Transmission Control Protocol) provides a way of sending larger amounts of data to particular 'ports' on a host.

TCP will resend missing packets and check that packets arrive in the correct order.

UDP: unreliable way; without a connection to be setup

Flip-flop

