CSSE 132 Exam 1
9-15-2018
Joshua Eckels

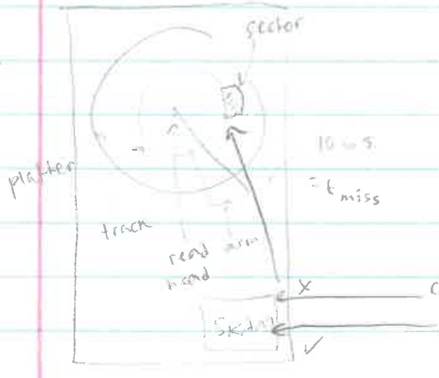
Final 11/12/18

|DE |AD |BE |EF → Big Endian

|EF |BE |AD |DE → Little Endian
↑ 1 Byte

- For 2^n bytes of memory, n bits are needed to address all the memory
Ex.- 2^3 Bytes → 3 bits

hard drive



10 ms = t_{miss}

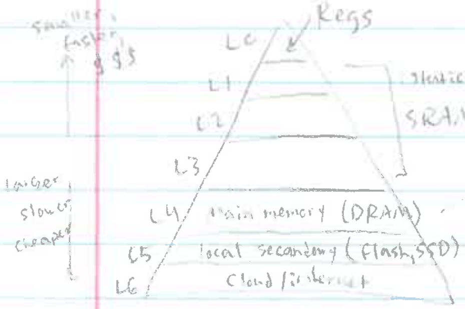
cache miss 00.01%
cache hit 99.99%
(500 ns)

$$t_{total} = P_{hit}(t_{hit}) + P_{miss}(t_{hit} + t_{miss})$$

$$= 0.9999(500ns) + 0.0001(500ns + 10ms)$$

$$t_{miss} = t_{seek} + t_{rotational\ latency} + t_{transfer}$$
(t_{access})

$$= (\sim 9ms) + \left(\frac{1}{2} rot \cdot \frac{sec}{rot}\right) + \left(\frac{1 rot}{\#sectors} \cdot \frac{sec}{rot}\right)$$



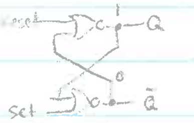
Static RAM)
SRAM (L1-L3) → caches (temporal, spatial locality)

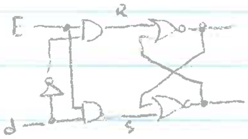
→ Dynamic RAM – capacitors; must be refreshed

SSD → read 1 page at a time / must write whole block / wears out with usage
↓
flash

Memory

SR latch



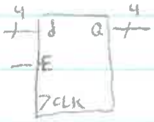


d latch w/ Enable

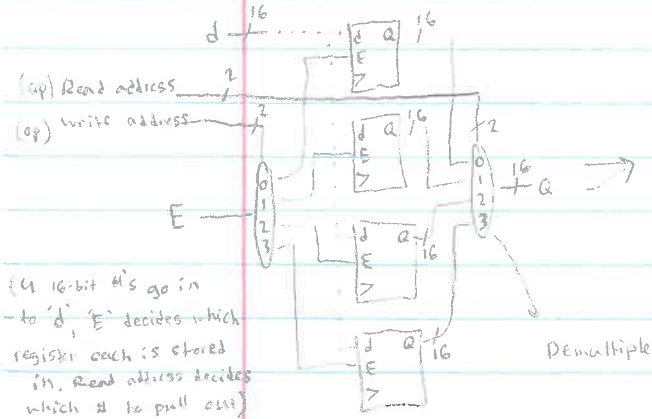
d	E	Q⁺
0	0	Q
0	1	0
1	0	Q
1	1	1

d flip-flop w/ Enable

4-bit register

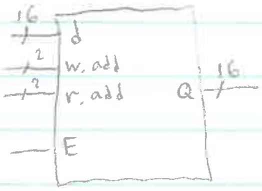


- d is copied to Q when E is a 1 and CLK is on rising edge



(4 16-bit #'s go in to 'd', 'E' decides which register each is stored in, Read address decides which # to pull out)

Register file



Demultiplexer:

Address	0	1	2	3
0 0	E	0	0	0
0 1	0	E	0	0
1 0	0	0	E	0
1 1	0	0	0	E

C_{in}
(1)
0 1 0 0
+ 0 1 1 0
C_o

Unsigned → addition → overflow if $C_o = 1$
↳ subtraction → overflow if $C_o = 0$
Signed → overflow if $C_o \neq C_i$

- ssh pi@eckelsjd-pi.wlan.rose-hulman.edu
- cd, ls, grep, head -n 173, tail -n 173, cat

/file name/file.txt → absolute path ; * → wildcard

head file.txt > newfile.txt (send to a file)
head file.txt >> newfile.txt (append to a file)

; grep "CLUE" filename.txt

head names.txt | grep "Steve"

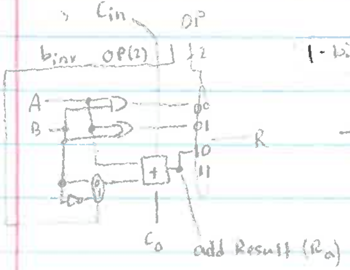
- "|" pipe sends output of a command to another command

grep -A, grep -B, grep -C
grep -I "Daffy" *

A = after, B = before
C = before and after
I = print file name

Cmd

$C_{in} = op(2)$
in multi-bit ALU
(only for LSB)

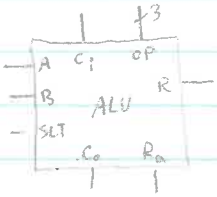


OP codes

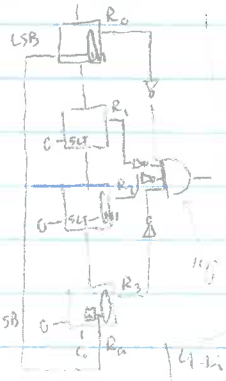
op(2)	op(1)	op(0)	
0	0	0	AND
0	0	1	OR
0	1	0	ADD
1	1	0	SUB
1	1	1	SLT

OP(2) → chooses SUB or ADD
(1) (0)

1-bit ALU



SLT of LSB = R_a of MSB
SLT = 0 for all other bits



"0" detector

4-bit ALU

(contents of a[0] copied into x)

server Socket connection:

int sd = socket()

bind (sd...

listen(sd...

accept (sd... → cs

recv (cs

send (cs

close (cs);

close(sd);

client:

int sd = socket();

connect(sd...

send (sd

recv(sd

close (sd

(r4-r11, lr, fp)

0 = stdin

1 = stdout

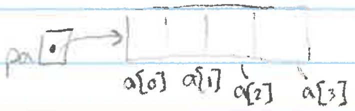
Pointers/Arrays

int a[4];

int *pa;

pa = &a[0];

x = *pa;



- pa+i points i elements after pa
- *(pa+i) refers to contents of a[i]
- pa = &a[0] → then pa(pointer) and a (array) have the same value
- &a[i] and a+i are identical addresses
- pa[i] is identical to *(pa+i)
- incrementing a pointer moves to next element in its array, so it increments by size of 1 element in array

Arm

sp → r13 → points to top of stack

fp → r11 → points to bottom of current frame

lr → r14 → holds return address

* str, ldr, mov, add, sub, cmp, b

.LOOP, .ENDLOOP, BEGINIF, .ELSE, .ENDIF

- steps for procedure to call another
  1. put args in r0-r3
  2. record return address in lr > bl
  3. go to procedure
- Steps a procedure takes when called
  1. back up registers needed on stack
  2. read args from r0-r3
  3. compute → put return value in r0
  4. restore backed up registers → b lr

unbuffered

write (fd, " ", # bytes)

read (fd, into array, # bytes)

open(location, flags)

lseek(fd, bytes from, the end of file)

lseek(fd, 0, SEEK_CUR)

returns location of cursor

buffered

- fwrite(data, size, count, stream)
- fread(*ptr, size, nmemb, FILE* stream)
  - reads from stream and stores at ptr
- feof(FILE* stream) → returns 1 at end of file
- ftell() → returns location of cursor
- fseek(FILE* stream, offset, location)
- fclose(file), fflush(file), fopen(file)

- int main(int argc, char** argv) {

  argc = # arguments

  argv = list of arguments (as strings)

  argv[0] = name of program

Struct db_entry stack;

stack.name = ...

stack.value = ...

		values	range
char	1 byte	2^8	$[-2^7, 2^7-1]$
int	4 bytes	2^{32}	$[-2^{31}, 2^{31}-1]$

1-bit 5-bit 10-bits

| s | exp | significand |

$1.0111_2 \times 2^{10_2}$ exp

sign significand

$1.0111_2 \cdot 2^{10_2} = 101.11_2$

$= 2^2 + 2^0 + 2^{-1} + 2^{-2}$

Unix

__ > file → redirects stdout

__ < file → redirects stdin

(1) | (2) → use output₁ as input₂

__ >> append output

find . -iname "*.c"

wc file

gcc -g -o kaboom kaboom.c

compile into

- man, pwd, cat, less
  head -n, tail -n, grep
- gdb server
- break server.c : 77
- run

- bridge - connect 2 like networks
- Router - " 2 unlike networks
- Network - group of connected computers
- Switch - central hub of a network