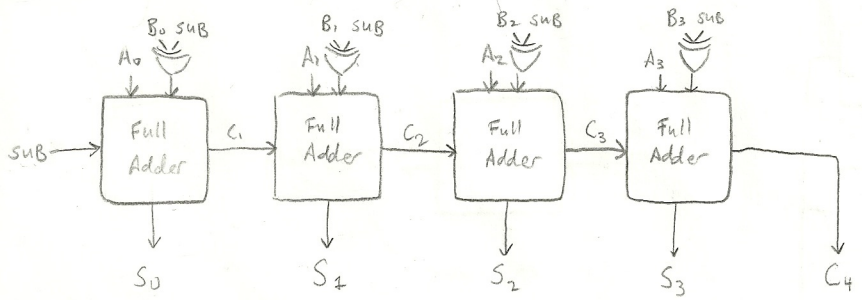
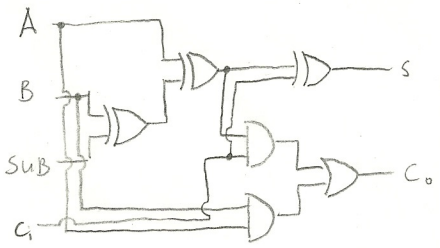
EE-357 :: HOMEWORK (MIDTERM REVIEW) :: ALEX YOUNG, ETHAN CHAN

(1)

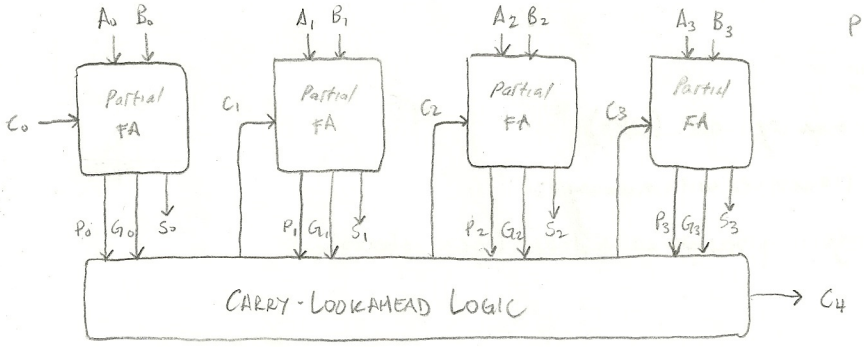


* For a 4 bit number
* SUB = 1 to subtract, SUB = 0 to add

Full Adder with Subtraction @ Gate Level:



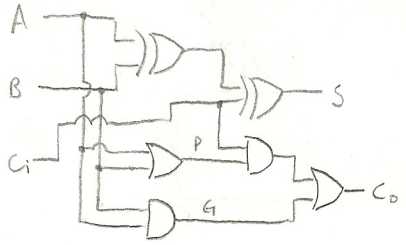
Runtime: O(n)

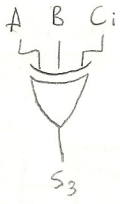


Partial Full Adder @ Gate Level:

$G = A \cdot B$ if C_o is generated

$P = A + B$ if C_i is passed to C_o

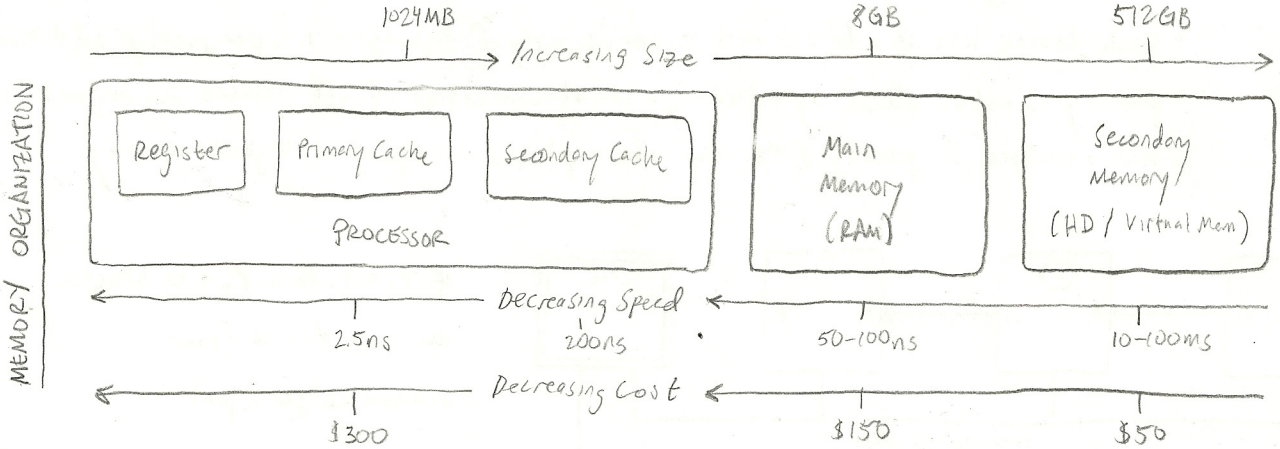




* forgot to add subtractor, just swap and make C_0 = SUB

AND $C_i = G_{i-1} + P_{i-1} C_{out\ i-1}$

(2)



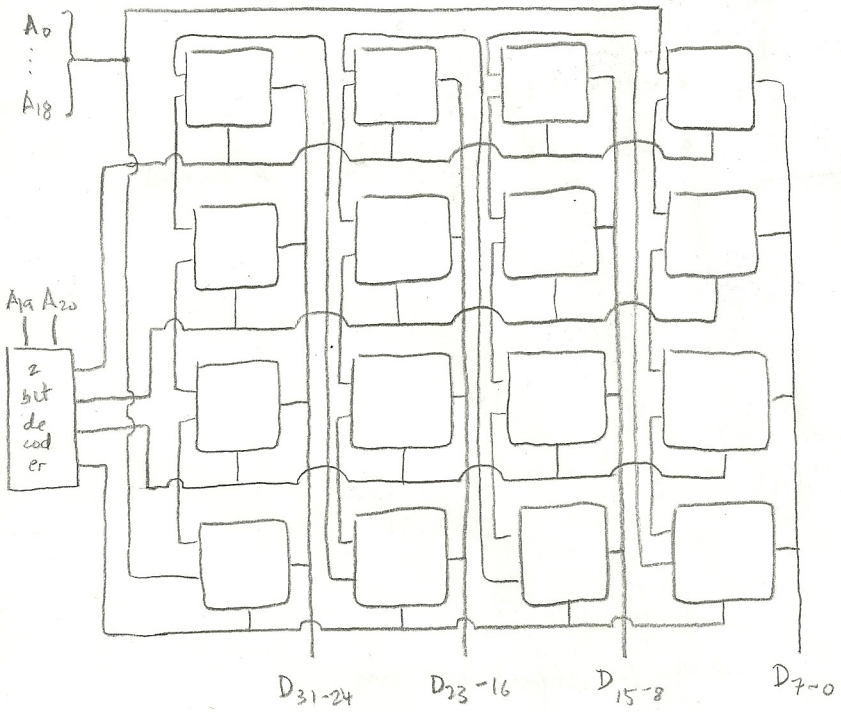
cache holds recently accessed information from RAM/HD to reduce fetching time and speed up processes

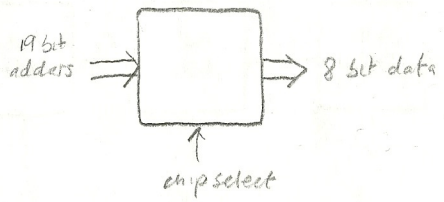
Three Methods:

1. Direct mapping: maps to a fixed position in the cache
2. Full association: maps to any position in the cache
3. Set association: can possibly exist in a range of locations in cache

→

Organization of a 2M×32 memory module using 512k × 8 static memory chip

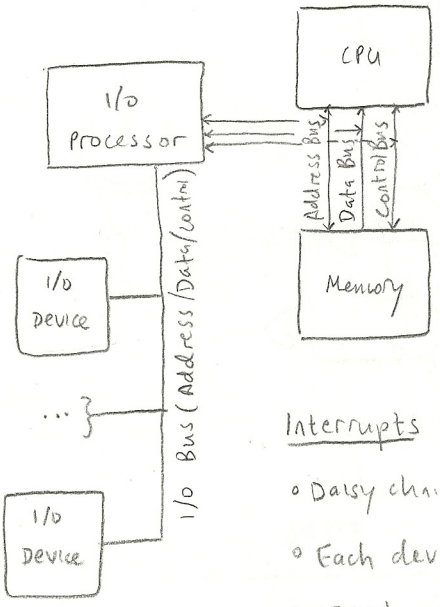




A19, A20 determine which rows are selected

R/$\overline{W}$ common connection

③

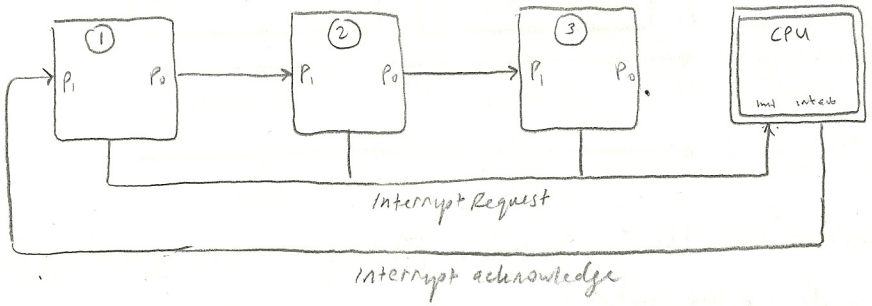


I/O Processor

- has direct memory access (DMA)
- combines I/O devices with memory
- replaces regular processor so it doesn't get interrupted

Interrupts

- Daisy chaining priority method is used to determine the source of the interrupt
- Each device has an address and responds only if the data is specifically addressed to it
- priority is established by including a serial connection of all devices the request interrupts
- halt an interrupt process if the new interrupt has an equal or higher priority



If $P_1 = 1$ then $P_0 = 0$ blocks the next device from acknowledge signal

(4)

```
LOAD num0, R0
LOAD num1, R1
CLEAR R2
CLEAR R3
BRA([R0] < 2^(n-1) - 1)
NOT R0
ADD R0, R0, #1
LOOP2 BRA([R1] < 2^(n-1) - 1)
NOT R1
ADD R1, R1, #1

BRA([R0] ≥ [R2])
STORE R2, DIVIDE
LOOP1
   SUB R0, R2, R0
   ADD R2, R2, #1
BRA([R0] ≥ [R1])
BRA([R4] ≠ 1)   LOOP3

NOT R3
ADD R3, R3, #1
LOOP3
   STORE R3, DIVIDE
```

LOOP2
ADD R3, R3, #1
LOOP1
ADD R3, R3, #1

MULTIPLY

```
LOAD NUM0, R0
LOAD Num1, R1
BRA([R1] < 2^(n-1) - 1) LOOP1
BRA([R0] < 2^(n-1) - 1) LOOP2
NOT R0
ADD R0, R0, #1
NOT R1
ADD R1, R1, #1
BRA LOOP1

LOOP1
   ADD R0, R0, R0
   SUB R1, R1, #1
   BRA([R1] > 0)

LOOP2
   SWAP R0, R1
   BRA LOOP1
```

(5) INSTRUCTION PIPELINE

a) Instruction fetch: instruction is fetched from RAM
b) Instruction decode: instruction is decoded
c) operand fetch: operands are fetched
d) execute: decoded instruction is executed
e) write back: result is written to processor register or memory

* Pipelining increases the number of instructions that can be sent at once

Instruction Number	PIPELINE STAGE						
1	IF	ID	OF	EX	WB		
2		IF	ID	OF	EX	WB	
3			IF	ID	OF	EX	WB
4				IF	ID	OF	EX
5					IF	ID	OF
clock cycle	1	2	3	4	5	6	7

→

(6) OUT-OF-ORDER EXECUTION

- processor can execute instruction in an order governed by availability of data, rather than their original order in a program
- if an operand is still unavailable during the clock cycle, the processor does not stall

Ex:

1) Instruction fetch
2) Instruction dispatch to instruction queue
3) Instruction waits in queue until its input operands are available. Instruction is then allowed to leave the queue before older instructions
4) Instructions are issued to appropriate functional unit and executed
5) Results are queued
6) Only after all older instructions have their results written back to the register file then is this result written back to the register file