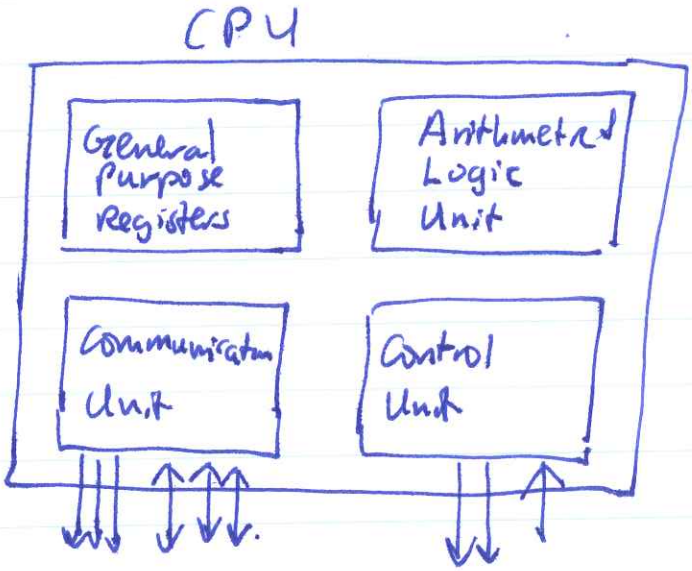
The CPU architecture

- CPU = central processing unit

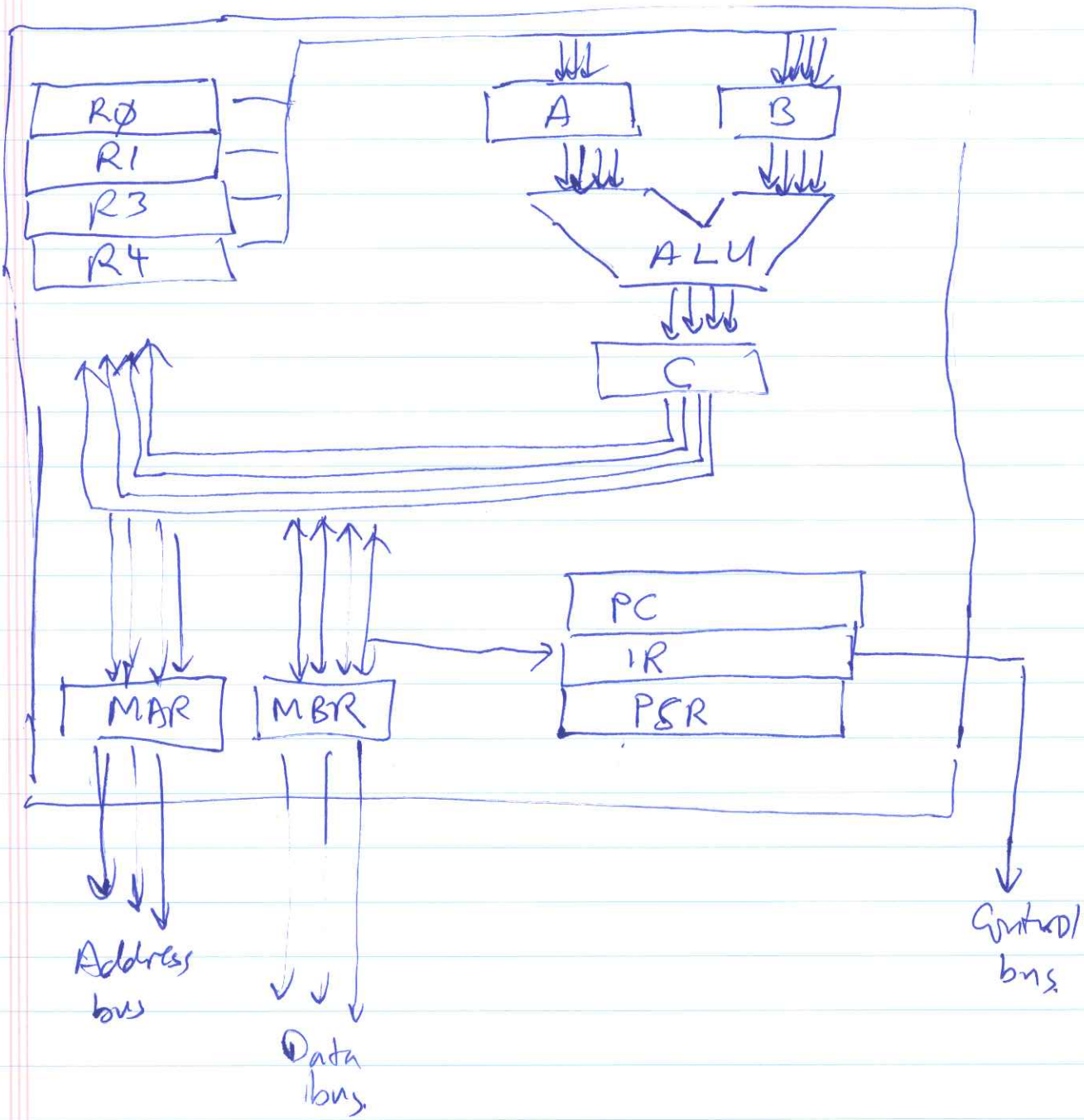
  Task: execute computer instructions fetched from memory.

- To fulfill this task, the CPU consists of 4 main parts:



- Register = very high speed memory.
- There are special purpose registers (registers that are used for a single purpose) in the ALU, Comm. Unit & Control Unit.

CPU overview :



General Purpose Registers.

- General purpose registers are very fast memory inside the CPU for storing intermediate results of computation.

  The general purpose registers can be loaded with values from memory, used in arithmetic / logic operations and written to memory.

  To compute a complex calculation, intermediate results are stored in registers.

  eg: $3+4*5$.

  is broken into 2 steps $4*5$ → store away
  then add.

- Each general purpose register has a unique name and is addressed by that name.

  eg. M68000 has 16 general purpose registers.

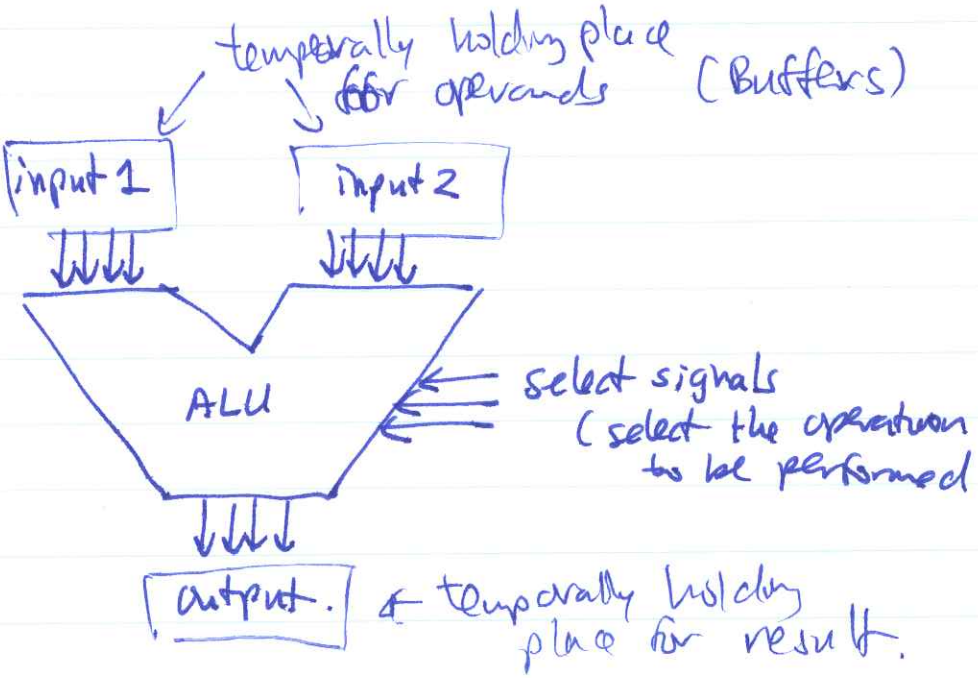
  - 8 Data registers named D0, D1, ... D7
  - 8 Address Registers named A0, A1, ... A7

Arithmetic & Logic Unit

- The ALU performs arithmetic (+, −, *, /) & logic (AND, OR, NOT, Exclusive OR etc) operations

- The ALU has 2 inputs and 1 output
  eg: to perform the operation 3 + 4, the values 3 & 4 are presented to the inputs and ALU is instructed (by the control unit) to perform addition. The result will appear on ALU's output.

Schematically:



logic-sim
demo/4-bit-ALU
00 = add
01 = not
10 = OR
11 = AND

- Where does the inputs ~~& output~~ come from?

  (1) General purpose registers.
  (2) from memory

- Output can go to register or memory.

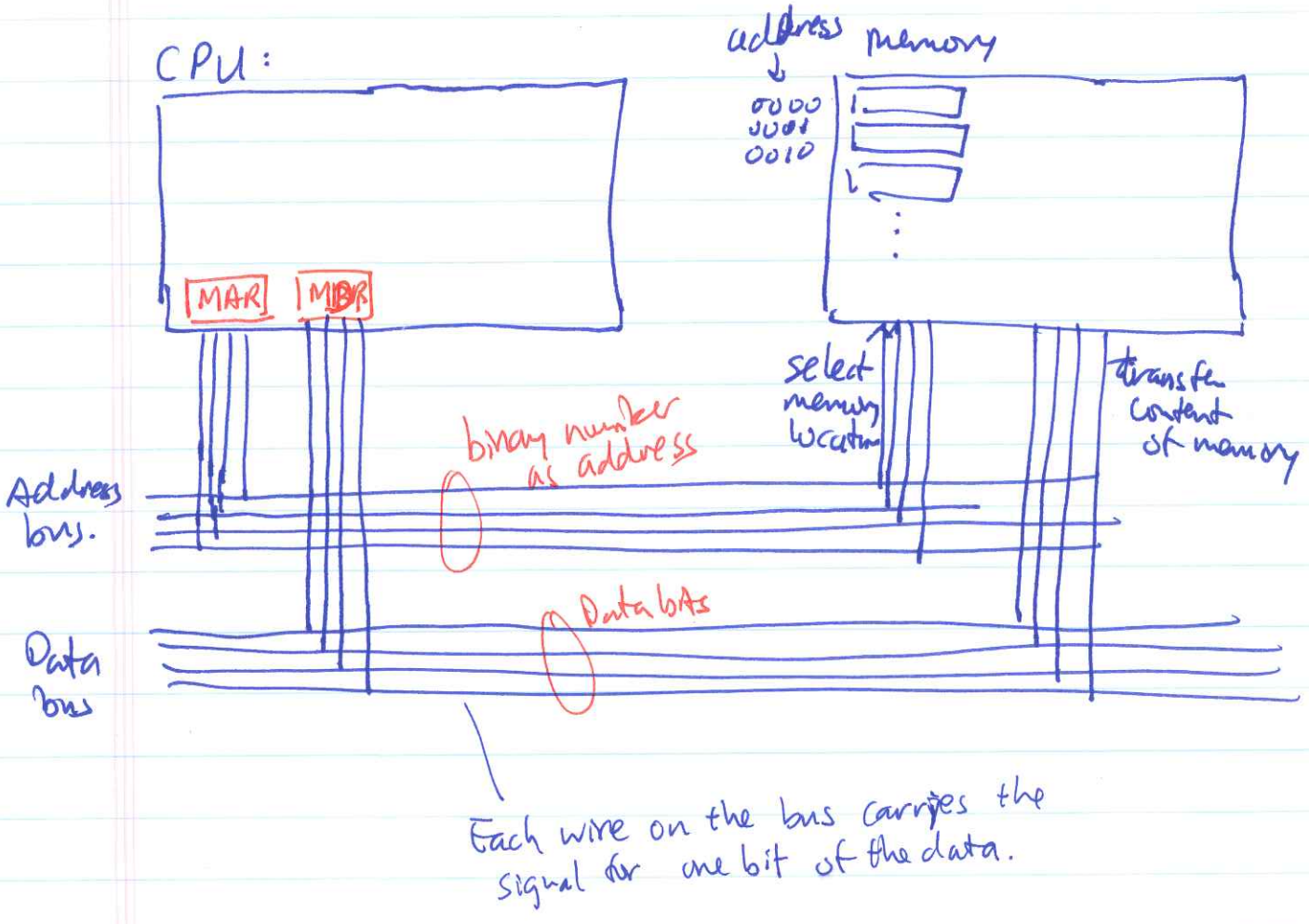
Communication with memory

- Recall that a memory location is identified by its unique address.

An address is expressed as an unsigned binary number.

The address is conveyed from the CPU to the memory through the address bus (bunch of electrical wires).

The data (content of the memory location) is transferred between the CPU & memory over the data bus.



- To make the communication with memory possible, the CPU's communication unit is equipped with 2 special purpose registers:

MAR = Memory Address Register
which contains the memory address that the CPU wants to access

The content of MAR is emitted on the address bus and propagated to the memory.

~~Upon reading the~~
Memory uses the value on the address bus to select the proper memory location.

MDR = Memory Data Register (also known as MBR).
which is used as a buffer (intermediate storage location) for data transfer between CPU & Memory.

During memory write operation where CPU wants to write a value to memory, the MDR is first loaded with the proper value, ~~& then~~
then MDR emits its value over the data bus to the memory.

During a read operation, memory sends the data over the data bus into the MDR.

The CPU will get the data a little bit later from the MDR.

- The length of the MAR determines the amount of memory the CPU can address.

eg. 8080 has a 16 bit MAR.

The amount of memory that the 8080 can address is

$$2^{16} = 64K \text{ bytes.}$$

M68000 has 32 bits MAR

(But usually not $2^{32} = 4$ G byte memory!)

- The length of the MDR determines the transfer rate, or transfer speed in # bits / transfer.

This can be compared to the # lanes on a highway: the more lanes, the high load of traffic can pass at any time.

Eg. M68000 has 16 bits data bus.
M68020, 68030 has 32 bits data bus.

80386DX	has	32	bits data bus
80386SX	has	16	bits data bus.
80486DX		32	
SX		16.	

What's the advantage?

80386DX can run almost twice as fast as the 80386SX because:

each instruction must be transferred (fetched) from memory to the CPU before it can executed.

With 32 bits data bus, instructions can be fetched twice as fast as a 16 bit data bus.

The Control Unit

- The control unit is responsible to coordinate the other components in order to accomplish the effect specified by the instruction.

Need information to perform the control.

The control unit has 3 special purpose registers:

(1) Program counter (PC) which contains the address of the next instruction in memory.

The instruction stored at the address given by the PC will be fetched into the CPU before execution begins.

(2) Instruction Register (IR) which contains the instruction that is being executed.

The control unit decodes the instruction given in IR and sends out the appropriate signals to other components (Registers, ALU & Comm. units) to achieve the effect specified by the instruction.

(3) Processor Status Register (PSR) which contains the status of the CPU.

Four flags (single bit memories) are of particular importance:

flag = bit in the PSR that is set to reflect the result of some previous operation.

N flag = set (1) if result is negative.
Z flag = set if result is zero
V = set if result produced overflow
C = set if result produced a carry or borrow.

---

eg. suppose in M68000 the registers D0 & D1 contains

D1 = 00...010 $(2)_{10}$
D0 = 00....011 $(3)_{10}$.

And the instruction

ADD D0,D1 (add D0 to D1 and store result in D1

(delay to later instructions (M68000 instructions).

After the execution

D1 = 00...0101 $(5)_{10}$

N = 0 (result not negative)
Z = 0 (result not zero)
V = 0 (result not overflow
C = 0 (no carry).

On the other hand, if

D# = 00 ... 010
D0 = 00 ... 011

and

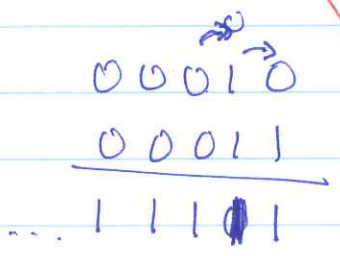
SUB D0, D1 was executed
(subtract D0 from D1
and store result in D1.

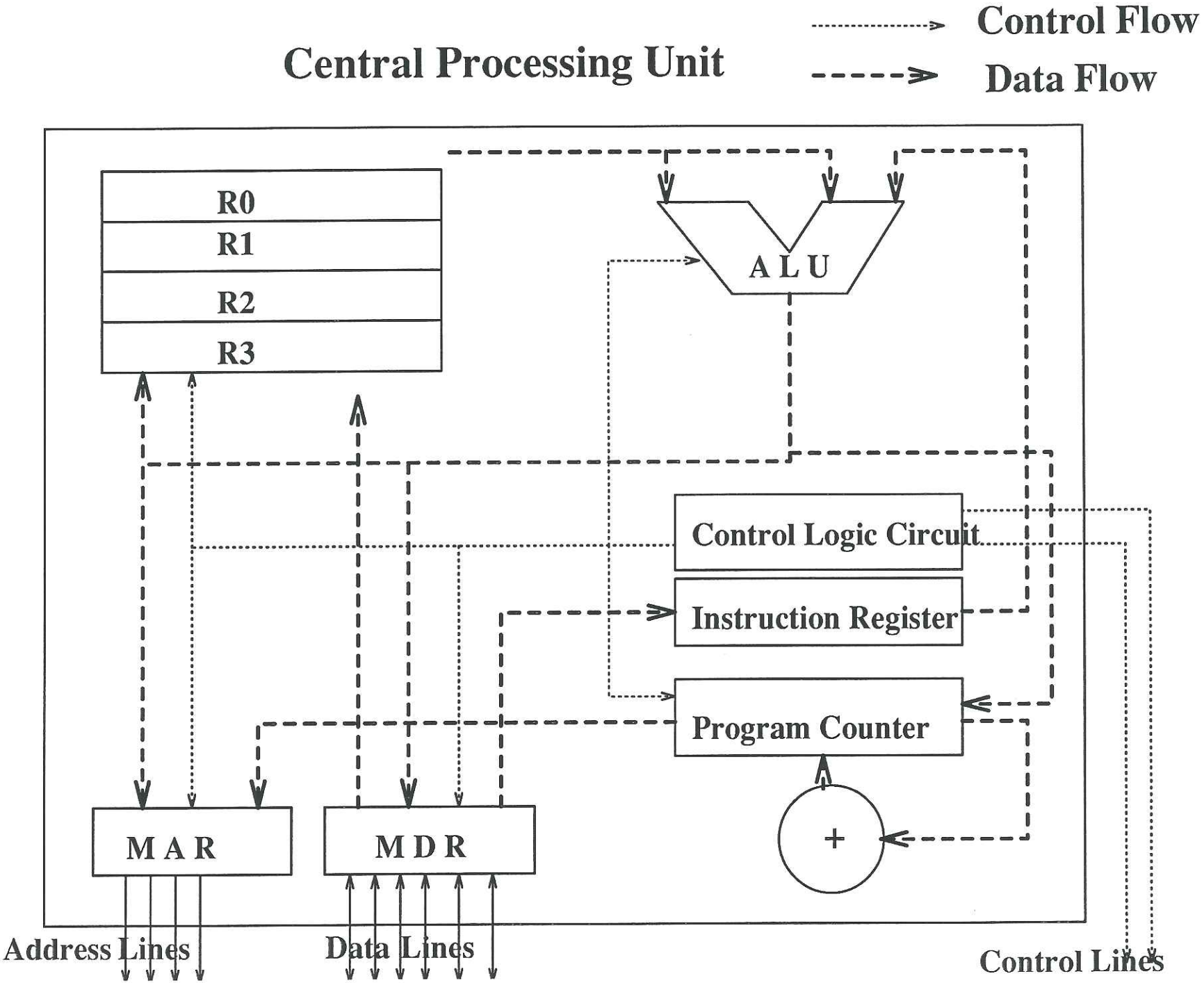
delay till later.

Then

D1 = 1111 ... 111 (−1).

and: N = 1 (negative)
Z = 0 (not zero).
V = 0 no overflow
C = 1 (borrow!)




Central Processing Unit
Control Flow
Data Flow
R0
R1
R2
R3
A L U
Control Logic Circuit
Instruction Register
Program Counter
+
M A R
M D R
Address Lines
Data Lines
Control Lines

Connection CPU, memory & I/O devices

- CPU – Memory, CPU – I/O devices, Memory – I/O devices are connected by "busses".

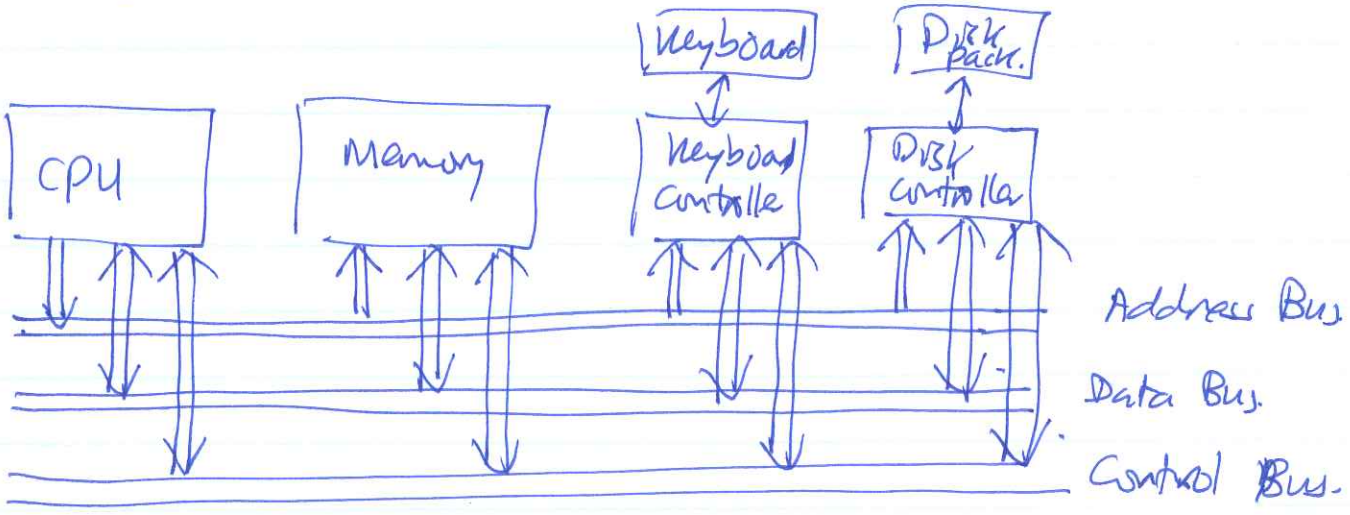
  A bus = a set of parallel wires.

  Each wire carries 1 bit of information.

- There are numerous ways to connect CPU – memory & I/O devices.

- The cheapest way is:

  All (CPU - memory - I/O devices) are on the same bus:



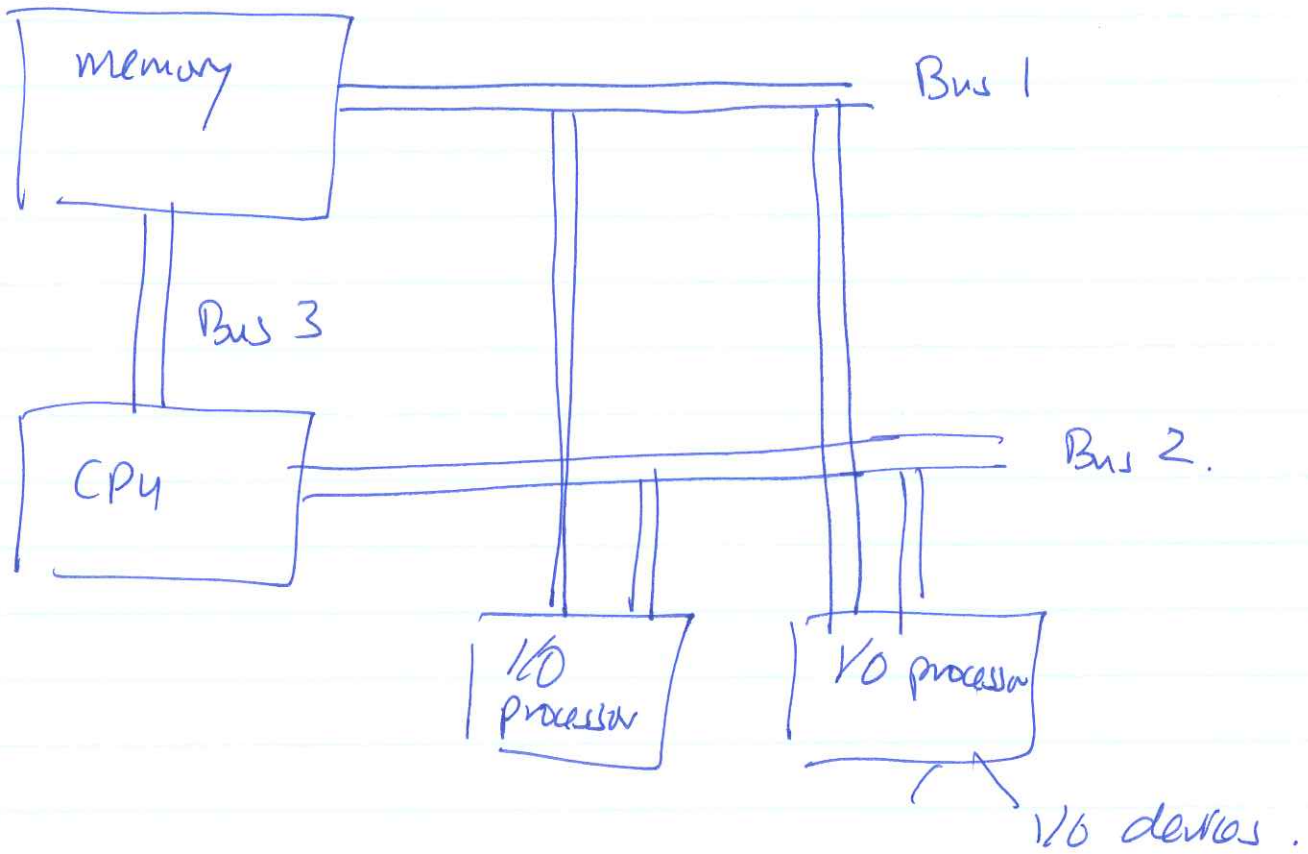
- This is how components in a PC are connected.

- Disadvantage:

  When CPU is fetching data / instruction from memory, I/O devices cannot use address / data buses to transfer data to from memory.

  (signals overlap!)

- More expensive: memory - CPU
  I/O device - CPU
  memory - I/O device
  } separate connections



(1) CPU access memory directly.
I/O processors also access memory directly.

(2) CPU instruct I/O processors to transfer data from / to memory.

- off load CPU from this task.

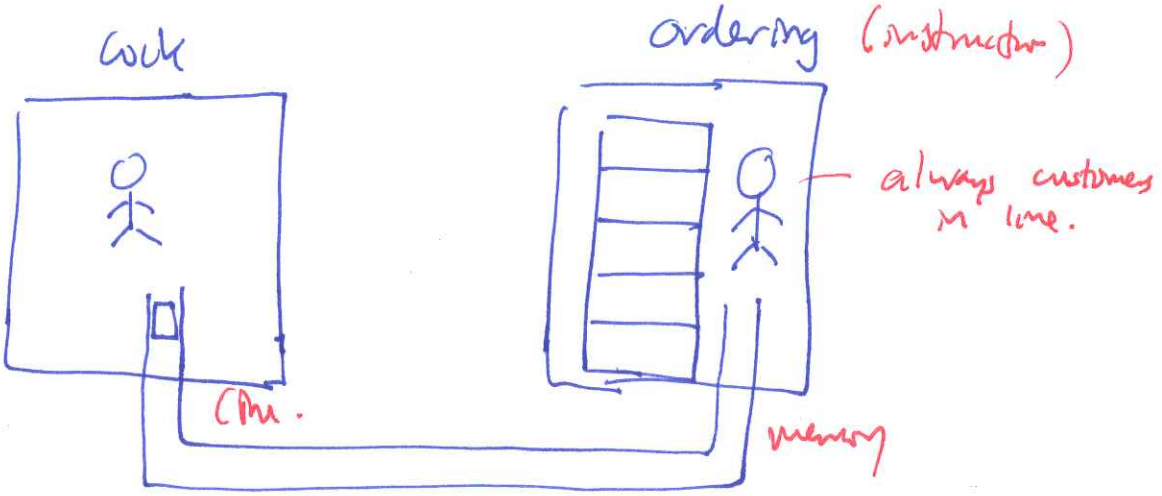
~~Also~~:

Major advantage:

CPU can fetch data & instruction from memory via bus 3 which I/O processor transfer data to/from memory via bus 1 simultaneously (unless BOTH CPU & I/O device use the same memory location - rare event).

Nice analogy about how CPU executes instructions:

Fast food restaurant:



- When the cook is ready, it gets the next order
- Communication is done through a tube with a qlinder.
- Ordering X-ERUX's the next order and send it to the cook.
- Cook process order and asks for more.
- Ordering ALWAYS has a next order for the cook to fill

The instruction execution cycle

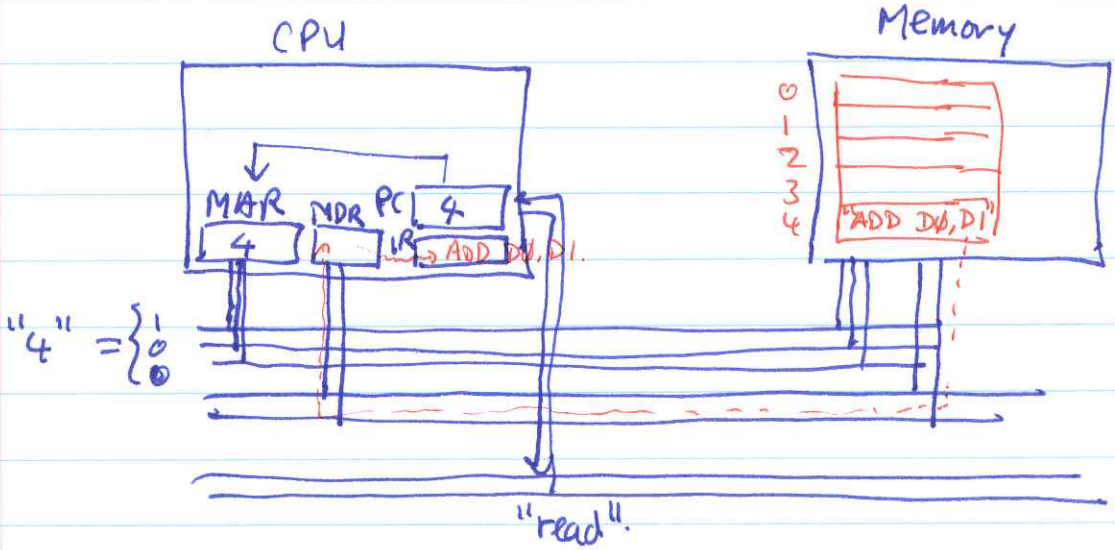
- The CPU performs the instruction execution cycle and instructions (for ever).

In one CPU cycle, one computer/machine instruction is fetched from memory into the CPU and executed.

Step 1: Instruction fetch

move contents of Program counter to memory adder. register ([MAR] ← [PC])

After transfering PC into MAR, PC is incremented ([PC] ← [PC] + "1". , actually: PC ← PC + length of instruction)



~~MAR now contains a copy of~~

CPU sends out the value in MAR over the address bus and instruct the memory to send it the content of that location.

Memory respond by sending the content of the selected memory to the MDR in the CPU. and then it is moved to the IR.

Result: the instruction code (bit pattern representing the instruction) at memory location PC is sent from memory to CPU's MDR and moved to CPU's IR.

This step is called "Instruction fetch".

Step 2: Instruction decode.

The instruction code, which is now in the IR, is divided into several fields:

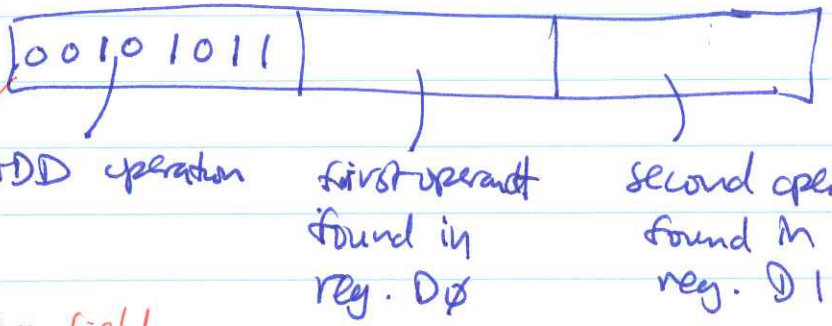
A field is a part of a word in which the bits are grouped together into a logical unit.

One field in the instruction code is the operation.

Other fields in the instruction code indicate the operands.

eg: ADD DØ, D1

The instruction code will have fields that encode:



Computer "looks" at this field and tries to determine what operation it must perform.

Note: how it is encoded depends on the computer designer.

Better explanation:

CPU determines from the operation code in the instruction what signals it must generate to ~~get~~ ~~then~~ execute the operation.

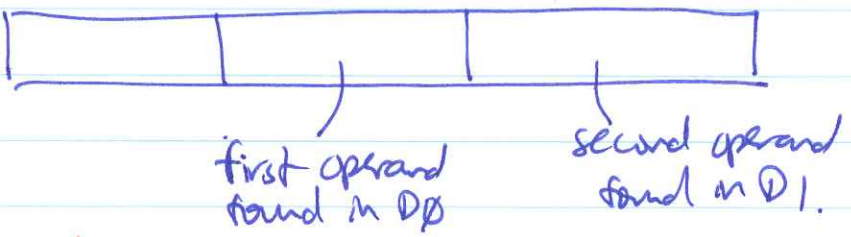
Step 3 : Operand Fetch

Before the operation (eg. add, subtract) can be executed, the value that the operation uses, must be fetched.

The control unit sends out commands to other parts of the computer to get the operands.

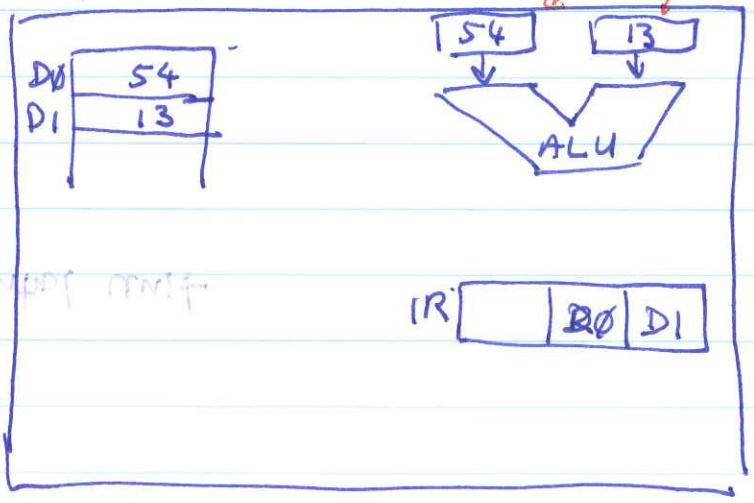
Where the operands are fetched from is determined by the bit pattern in the operand fields of the instruction code in IR.

eg: ADD D0, D1



operands fetched are put in special purpose input registers of the ALU

special purpose input register of ALU.



The operands are always brought to the two inputs of the ALU.

Once the inputs are ready, the operation can be performed.

Step 4: Execute.

The control unit sends out signal to the ALU ~~out~~ to do the operation.

What signals are sent is determined by the operation code part of the instruction code.

~~The result from the ALU is directed to its target.~~

~~The target is determined~~ also determined ~~by~~ by the 2nd operand.

In our example (M60000), the target is determined by the 2nd operand.

(In other CPU's a third operand field may be used).

Step 5: Write back:

The output of the ALU is fed back to either a general purpose register or to MBR which is then transfered to memory.

= Repeat Instruction cycle again ... and again ... and again ...

- NOTE: Recall that computer executes 1 instruction in an instruction execution cycle.

From what we have seen, a single computer instruction "does not do much", i.e., it only do a simple task.

~~Commonly~~

Normally, a machine instruction does the following:

- get a word/byte from memory into a register.
- put the word/byte in a register to some memory location.
- Add/Subtract/*/÷ 2 quantities.
- etc.

How computers execute a program in C or Pascal

- C or Pascal is a high level language used to express algorithms.

Statements in C or Pascal are TOO COMPLEX to be executed by any computer.

Each statement must be TRANSLATED into a number of simple steps that accomplish the EFFECT of the statement.

Example: `A = B + C;`

Is translated to:

```
MOVE   B, DØ      (get B in register DØ)
ADD    C, DØ      (add C to registe DØ)
MOVE   DØ, A      (put register DØ in A)
```