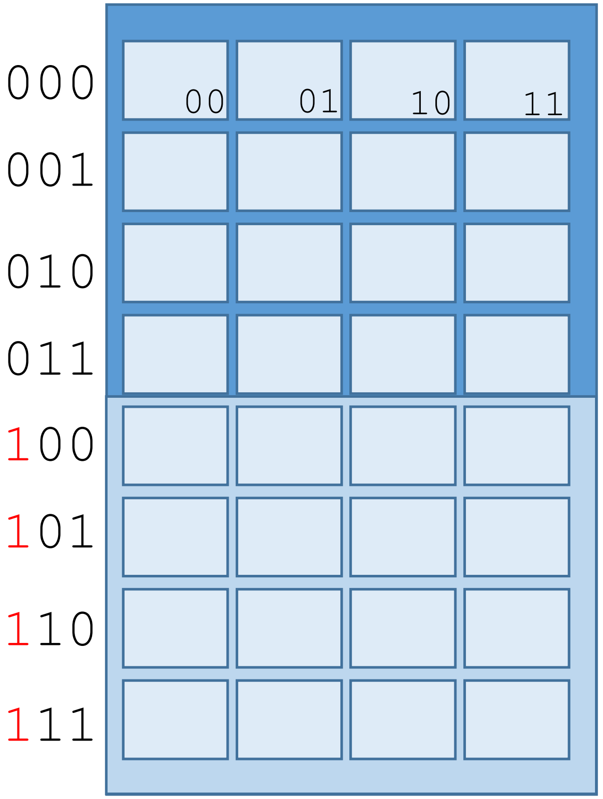
Microprocessor Architecture
Memory address space

Session 2.3

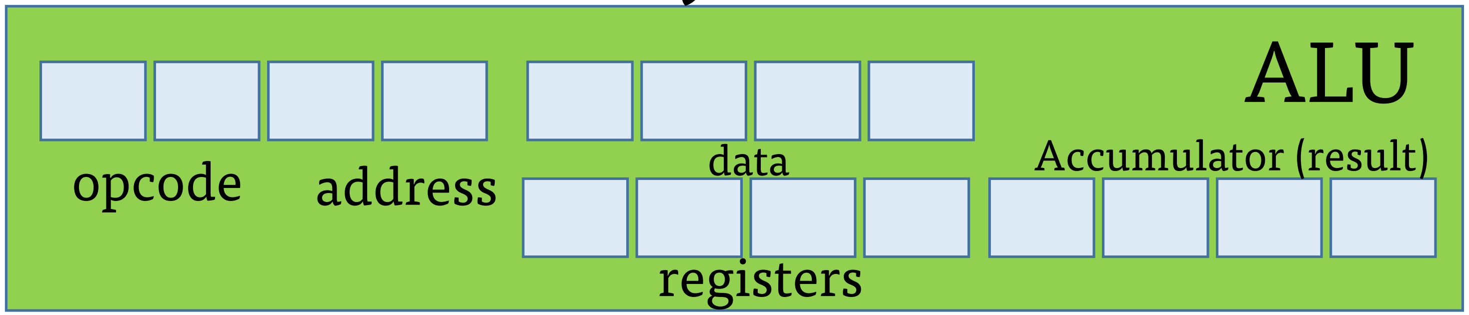
PH435

Reprise from Session 2.2



We started discussing the ENIAC designed at Princeton by J. von Neumann

We designed this ALU (CPU) from the bottom-up, carefully separating prog memory from data memory



add, subtract, read, write
Opcodes: `00,01,10,11`

This ALU is hardware programmed i.e. its **firmware**[D.G, S.T] is setup to do the following:

```
At power on, execute opcode at program address 0_00
```

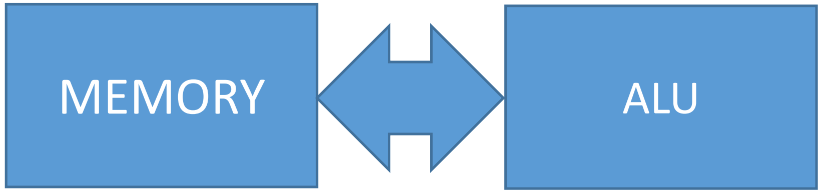
Our “toy” microprocessor runs a program

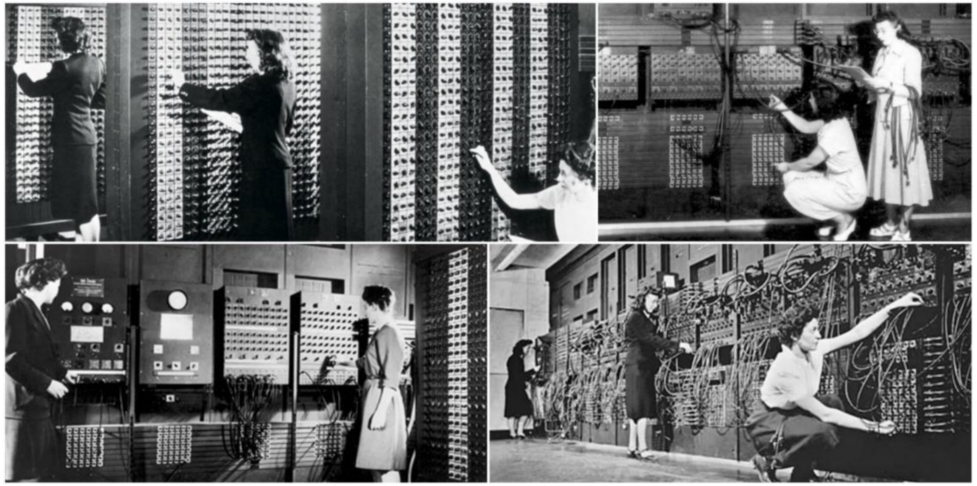
000	1	0	0	0
001	1	0	0	1
010	0	0	0	1
011	1	1	1	0
100				
101				
110				
111				

Homework:

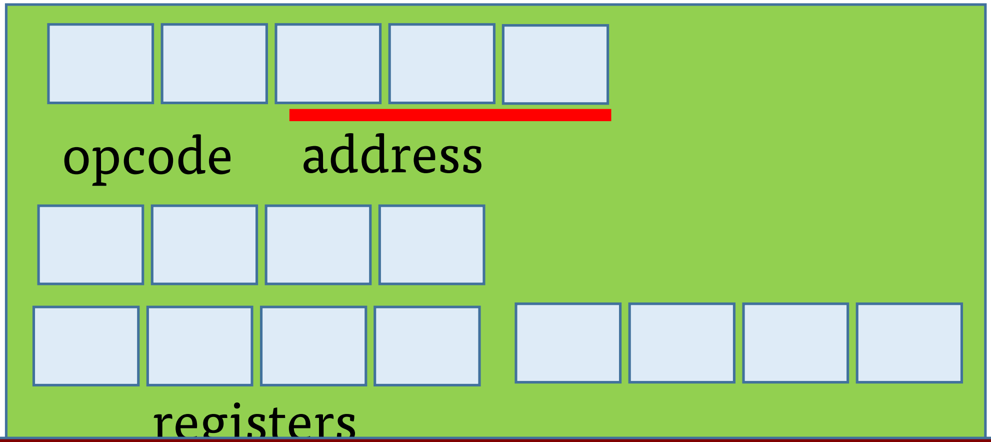
1) (SELF-STUDY) Decompile – trivial! addition

2) **HACK: [R.S.] if we had not been careful separating program memory from data memory address space at the design level, it would be possible to replace** 11_110→11_000





ENIAC (Princeton) was *designed* to have all memory accessible to CPU common over a common bus due to physical constraints: all the cabinets filled with memory banks were on one side of the room and ALU was on the other side.



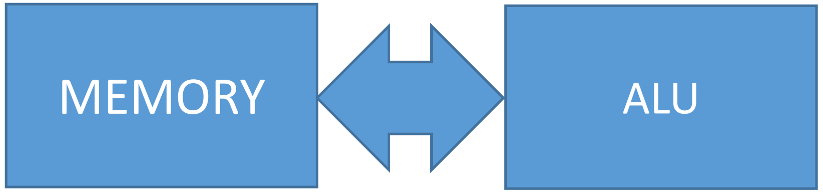
Concept of address spaces

000	1	0	0	0
001	1	0	0	1
010	0	0	0	1
011	1	1	1	0
100				
101				
110				
111				

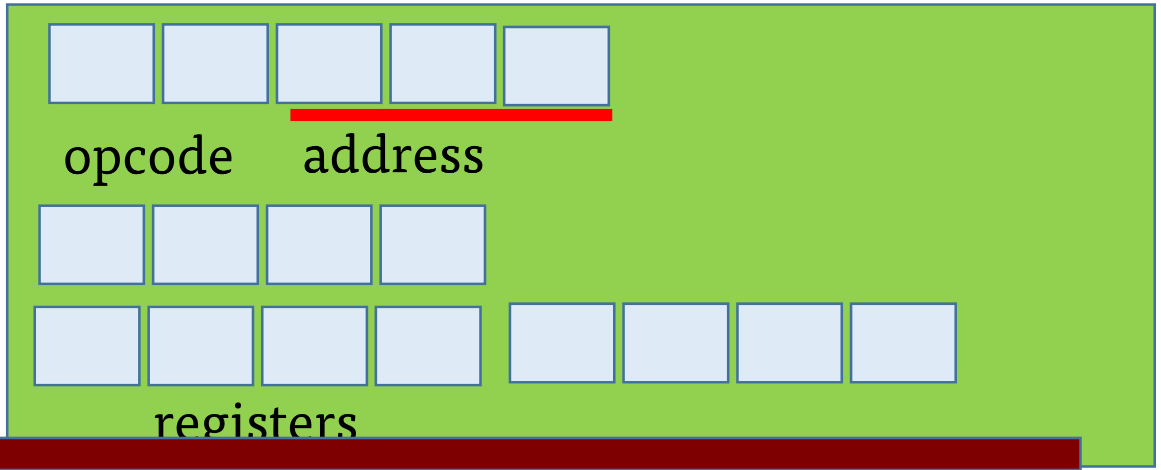
Our memory ‘IC’ breadboard has 8 addresses, each stores a 4-bit data

In principle, the ALU should be able to access all 8 addresses (3-bit address)

We artificially partitioned the memory



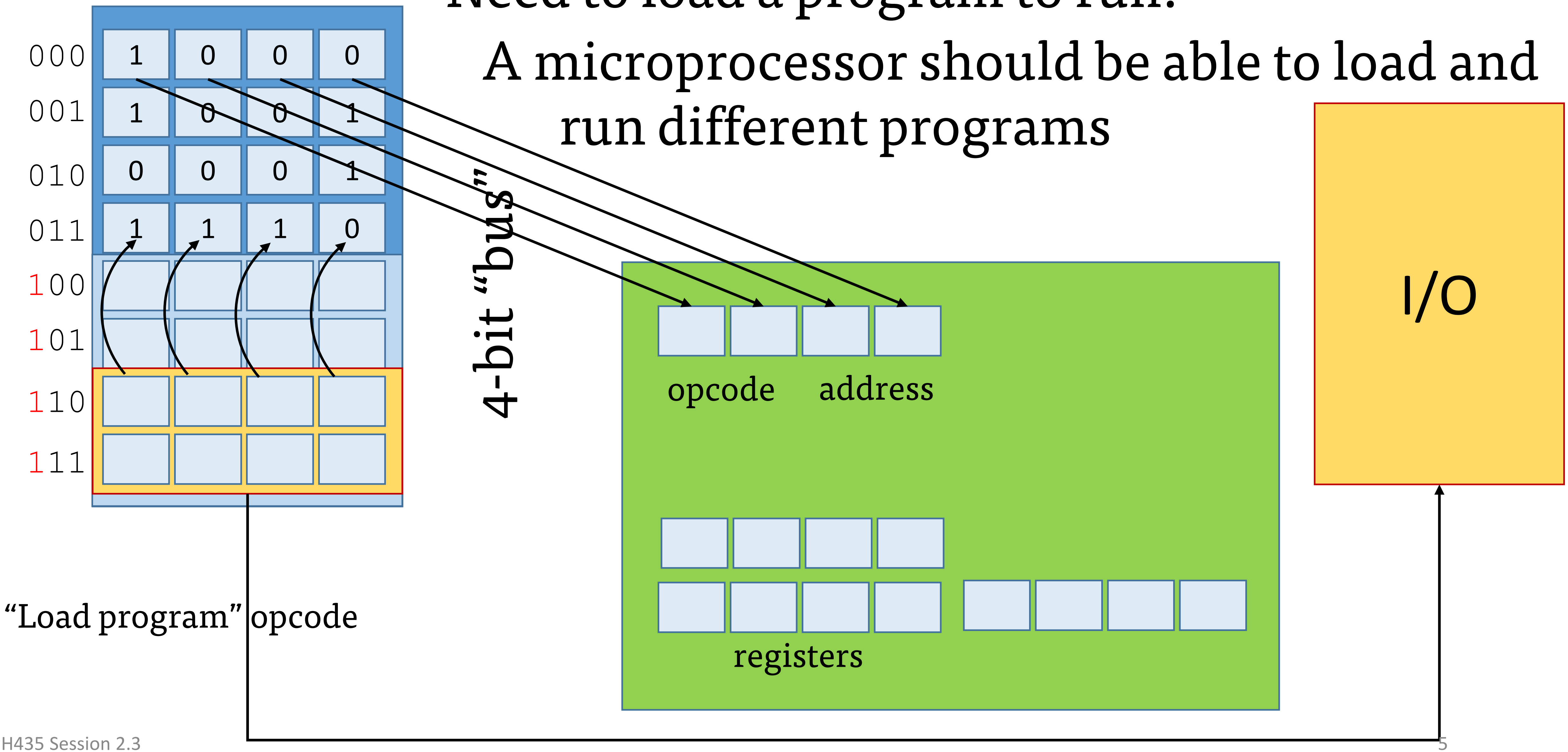
Also, only allowed 4 opcodes (2 bits) Should have more opcodes – eg JUMP to address etc…



Why does all memory need to be accessible?

Need to load a program to run! ~~[R.S.]~~

A microprocessor should be able to load and run different programs



Key takeaways:

Every microprocessor has a RESET vector address
At power up, opcode present at RESET vector address
is loaded and executed

<u>Address space</u>:
1) program address (where program code is read)
2) data address space (where volatile data resides)

<u>I/O address space</u> – some part of memory address space reserved for I/O.

Examples: a key press is recorded (as per key code) at that memory location and read by next opcode
Results of a calculation by CPU written out to that memory address and displayed to user. In case of ENIAC : use a card punch machine → Arduino: drive a digital o/p line H/L (LED))

Key takeaways:

Every microprocessor has a RESET vector address

At power up, opcode present at RESET vector address is loaded and executed

RESET VECTOR:
[prog.memory address space],[prog.memory address]

So far we have done a ‘thought’ experiment constructing a toy microprocessor and poking around its architecture

Princeton v/s Harvard Architecture

Speed limit – Dennard Scaling

Alternate computing model
Neuromorphic ‘in memory’