CRICOS PROVIDER 00123M

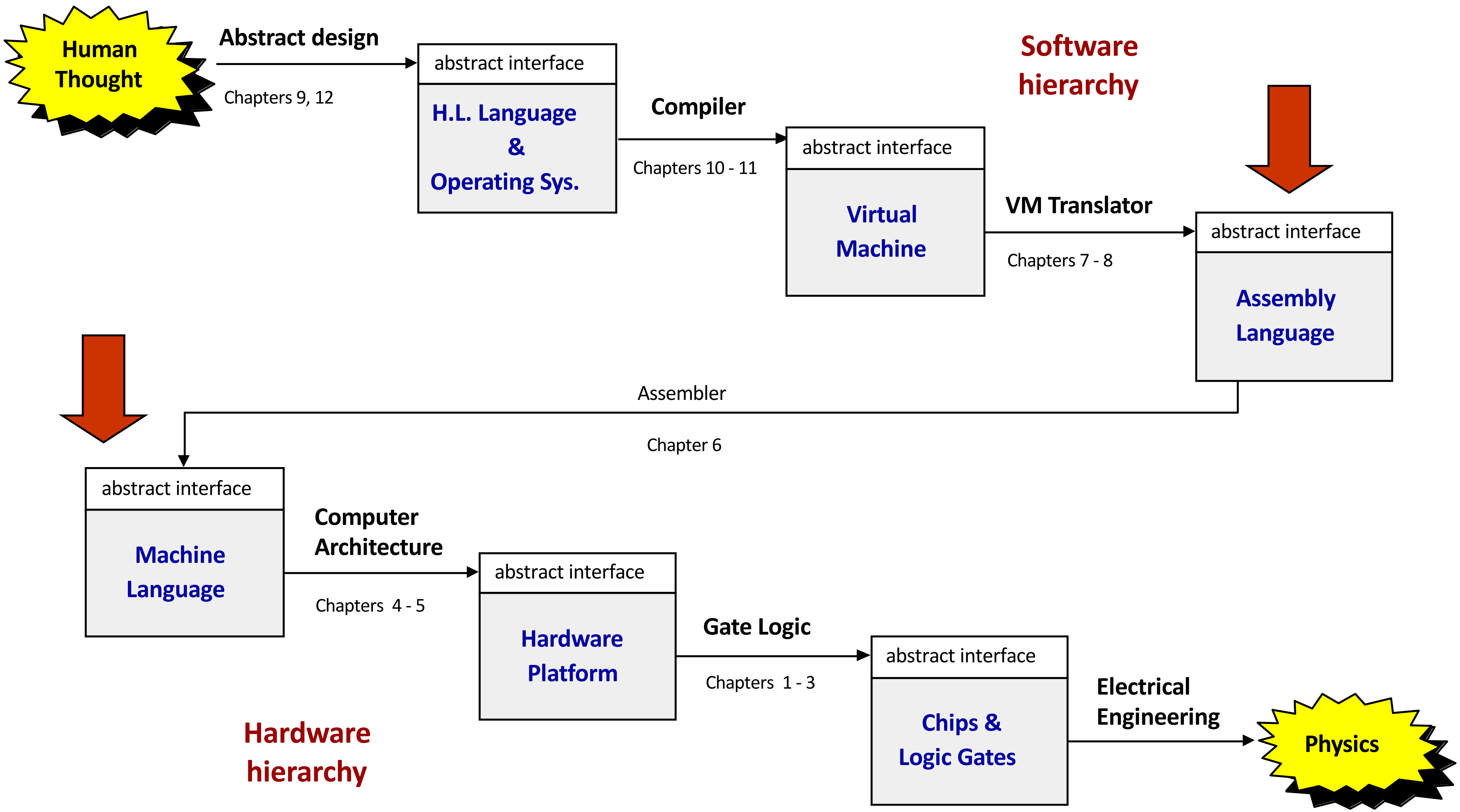
School of Computer Science

COMP SCI 2000 Computer Systems
Lecture 6

Review – RAM

- RAM is built up from smaller elements.
- We use address information to extract the correct element of RAM
- Caching strategies allow us to balance cost and access speed.
- Although everything *can* be built out of NAND gates, we generally optimise memory structures for performance.

Where we are at:



What we're doing now

- This week we're going to talk about:
  - Programming languages
    - Quick overview
    - History
    - High level languages can be very similar (or near identical) across different hardware platforms.
  - Machine languages
    - Manipulating memory using a processor and a set of registers
    - Will be different across different hardware platforms
  - Machine language in HACK

Early Programming

- Early machines
  - Jacquard loom
  - Automata
    - Physical coding on cards to trigger mechanical action
- Early calculators
  - Babbage's Difference Engine
    - Not programmable in the true sense.
    - Mechanical and analogue calculator
  - Babbage's Analytical Engine
    - Programmable for calculation
    - First real 'program' for Bernoulli numbers by Ada, Countess of Lovelace. (Never actually ran but would have worked.)

Early Languages

- Mathematical!
- Plankalkül from 1944 for the Zuse Z3

```
P1 max3 (V0[:8.0],V1[:8.0],V2[:8.0]) → R0[:8.0]
max(V0[:8.0],V1[:8.0]) → Z1[:8.0]
max(Z1[:8.0],V2[:8.0]) → R0[:8.0]
END
P2 max (V0[:8.0],V1[:8.0]) → R0[:8.0]
V0[:8.0] → Z1[:8.0]
(Z1[:8.0] < V1[:8.0]) → V1[:8.0] → Z1[:8.0]
Z1[:8.0] → R0[:8.0]
END
```

- Required expert knowledge and often didn't abstract very far at all.
- "English" programming didn't appear until 1955 with Vice Admiral Grace Hopper and FLOW-MATIC.

The effect of HLLs

- A high-level language has to be translated to a machine-interpretable form prior to execution.
- Translation often turns the HLL into an intermediate language that is then translated to machine code.
  - Why is there an intermediate step?
- HLLs are highly readable and allow a great deal of abstract representation
  - Each statement may be translated into many machine code instructions.

Do Lecture 6
worksheet
questsion 1

The ancestry of C++

- C++ (1980)
- C (1972)
- B (1969)
- BCPL (1967)
- CPL (1963)
- ALGOL 60 (1960)
- ALGOL 58 (1958)
  - Incidentally, the language that BNF was developed to define.

Do Lecture 6
worksheet
questsion 2

Thinking about Language and Hardware

Abstraction – implementation duality:

- Machine language (instruction set) can be viewed as a programmer-oriented abstraction of the hardware platform

- The hardware platform can be viewed as a physical means for realizing the machine language abstraction

Machine Language

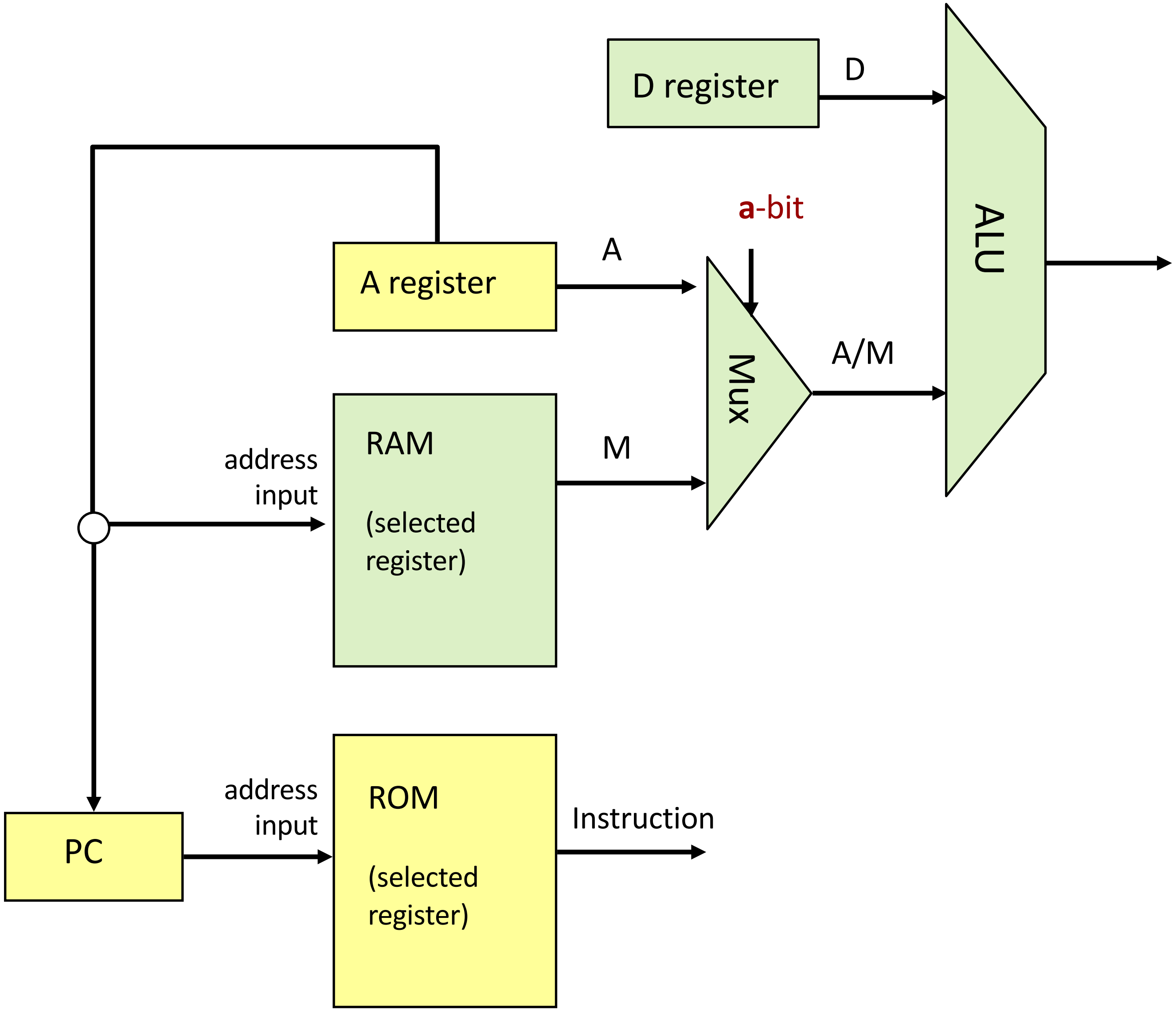
- Although the hardware is different, the idea of what a machine language does is the same from machine to machine.
  - an agreed-upon formalism for manipulating memory using a processor and a set of registers
- If you understand how one machine language works, you can understand (almost) all of them.
  - This assumes Von Neumann architectures!

What’s the general architecture?

- In small groups, sketch out a simple CPU.

Do Lecture 6
worksheet
questsion 3

What's the general architecture?



What instructions do we *need*?

What instructions do we *need?*

- Mathematical operations
- Logical operations
- Flow of control instructions
- Conditions
- Loading into memory
- Reading from memory
- Do nothing
- …
  - Examples?

Typical machine language (a small sample)

```
// In what follows R1,R2,R3 are registers, PC is program counter,
// and addr is some value.

ADD R1,R2,R3       // R1 ← R2 + R3

ADDI R1,R2,addr    // R1 ← R2 + addr

AND R1,R1,R2       // R1 ← R1 and R2 (bit-wise)

JMP addr           // PC ← addr

JEQ R1,R2,addr     // IF R1 == R2 THEN PC ← addr ELSE PC++

LOAD R1, addr      // R1 ← RAM[addr]

STORE R1, addr     // RAM[addr] ← R1

NOP                // Do nothing

// Etc. – some 50-300 command variants
```

Thinking about commands

- Why do we need an ADD *and* an ADDI?
- Do you know what RISC and CISC are?
  - Why is this important?

Do Lecture 6 worksheet questsion 4

Next Lecture

- Machine language programming in Hack!