I/O Systems

Design of Digital Circuits 2017
Srdjan Capkun
Frank K. Gürkaynak

http://www.syssec.ethz.ch/education/Digitaltechnik_17

Adapted from Digital Design and Computer Architecture, David Money Harris & Sarah L. Harris ©2007 Elsevier

What Will We Learn?

- **How to Interface Peripherals**
- **Memory Mapped I/O**
- **Memory Mapped I/O Example**
  - Speech Chip (SPO 256)

What Can Our Processor Do So Far?

- **We can:**
  - Calculate
  - Write to and Read from memories
  - Branch, loop, jump and return to sub-programs
- **But how do we interact with the environment?**
  - Light the LEDs on the board
  - Read the buttons
  - More complex interfaces, keyboards, video connections

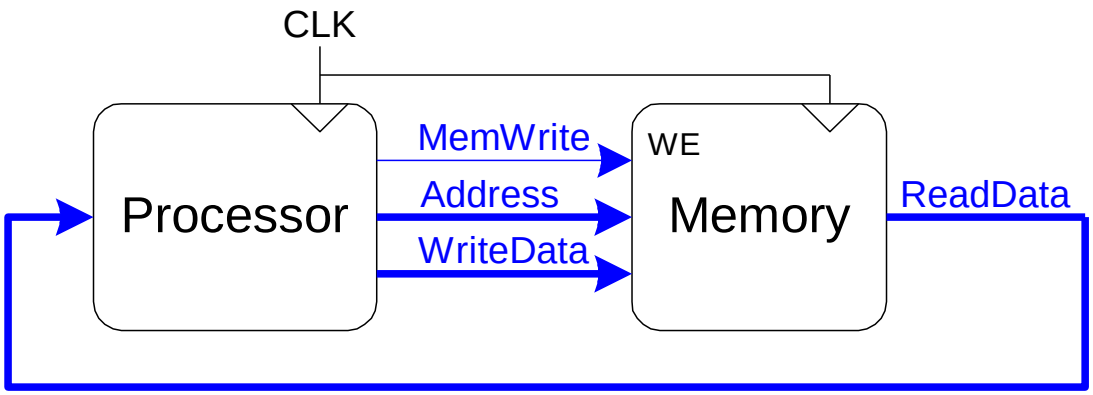
Memory-Mapped Input/Output (I/O)

- **Access I/O devices (like keyboards, monitors, printers) just like it accesses memory**
  - Each I/O device assigned one or more address
  - When that address is detected, data is read from or written to I/O device instead of memory
  - A portion of the address space dedicated to I/O devices (for example, addresses 0xFFFF0000 to 0xFFFFFFFF in reserved segment of memory map)
- **But we need additional hardware to help us**
  - After all we will not really write to and read from memory

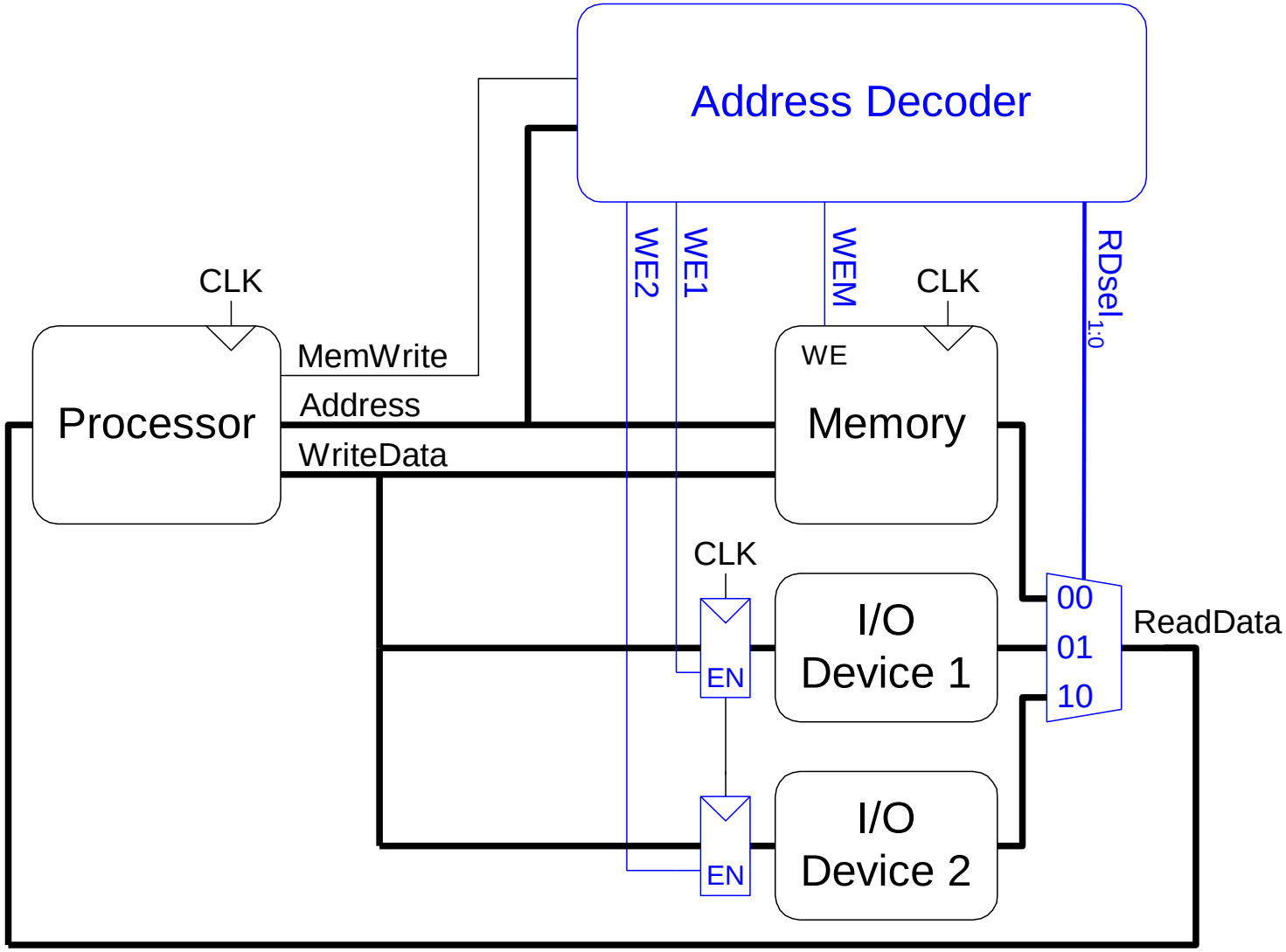
Memory-Mapped I/O Hardware

- **Address Decoder:**
  - Looks at address to determine which device/memory communicates with the processor
- **I/O Registers:**
  - Hold values written to the I/O devices
- **ReadData Multiplexer:**
  - Selects between memory and I/O devices as source of data sent to the processor

The Memory Interface



Memory-Mapped I/O Hardware



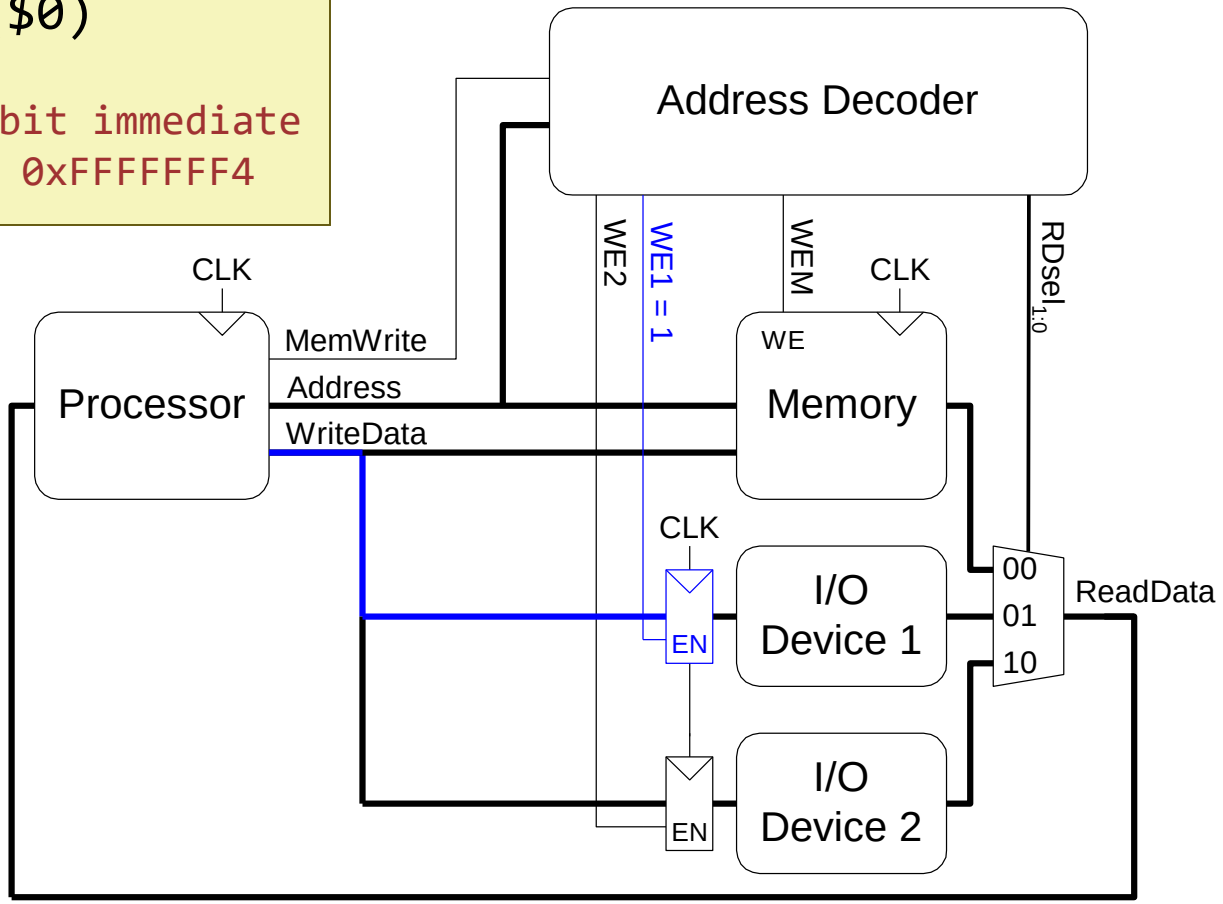
Memory-Mapped I/O Code

- **Suppose I/O Device 1 is assigned the address 0xFFFFFFF4**
  - Write the value 42 to I/O Device 1
  - Read the value from I/O Device 1 and place it in $t3

Memory-Mapped I/O Code: Write

Write 42 to I/O Device 1 (0xFFFFFFF4)

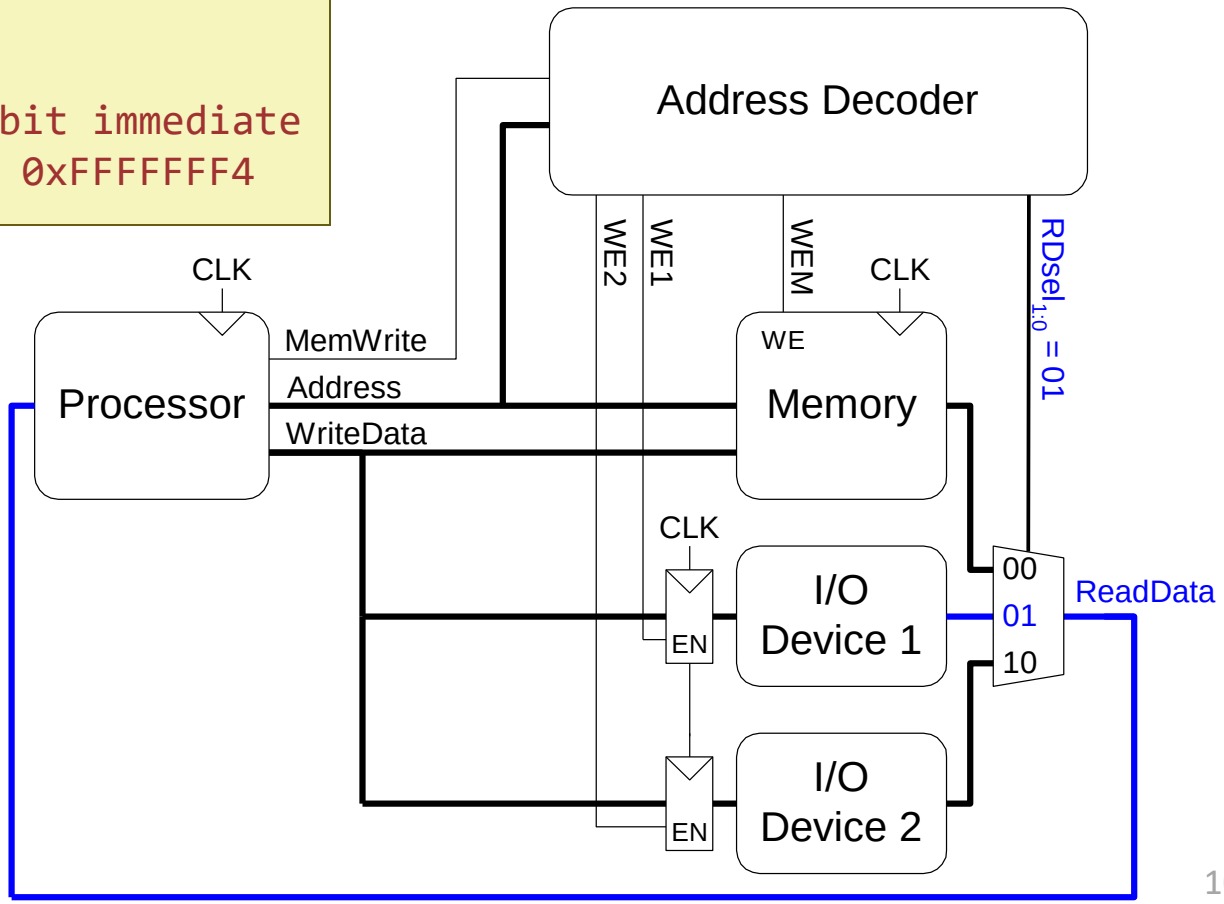
```
addi $t0, $0, 42
sw   $t0, 0xFFF4($0)

# Recall that the 16-bit immediate
# is sign-extended to 0xFFFFFFF4
```



Memory-Mapped I/O Code: Read

Read from I/O Device 1 and place it in $t3

```
lw $t3, 0xFFF4($0)


# Recall that the 16-bit immediate
# is sign-extended to 0xFFFFFFF4
```

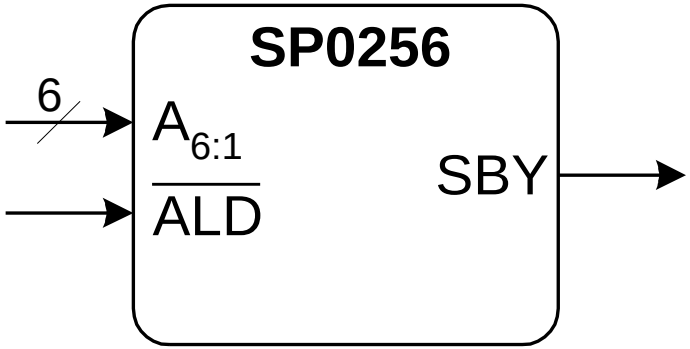


Example I/O Device: Speech Chip SPO256

- **Allophone: fundamental unit of sound, for example:**
  - "hello" = `HH1 EH LL AX OW`
- **Each allophone assigned a 6-bit code, for example:**
  - "hello" = `0x1B 0x07 0x2D 0x0F 0x20`

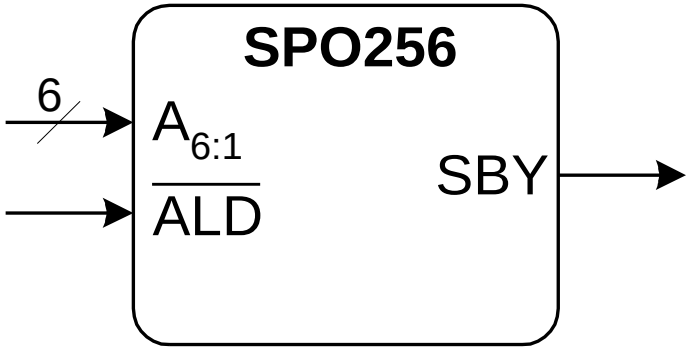
See http://www.speechchips.com, http://www.futurebots.com/sensor.htm

Speech Chip I/O



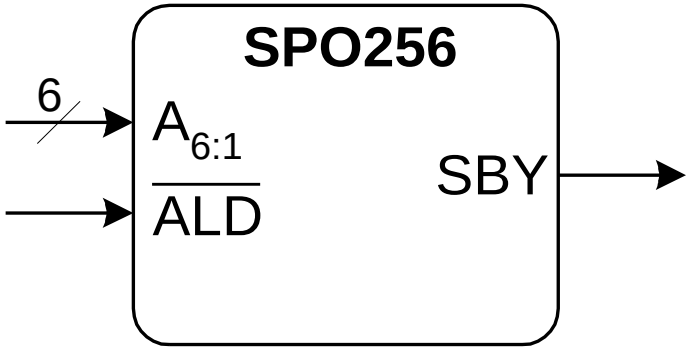
- **$A_{6:1}$: allophone input**
- **$\overline{ALD}$: allophone load (the bar over the name indicates it is low-asserted, i.e. the chip loads the address when $\overline{ALD}$ goes low)**
- **SBY: standby, indicates when the speech chip is standing by waiting for the next allophone**

Driving the Speech Chip



- **Set $\overline{ALD}$ to 1**
- **Wait until the chip asserts SBY to indicate that it has finished speaking the previous allophone and is ready for the next one**
- **Write a 6-bit allophone to $A_{6:1}$**
- **Reset $\overline{ALD}$ to 0 to initiate speech**

Memory-Mapping the I/O Ports



Memory Mapped I/O

$A_{6:1}$: `0xFFFFFF00`

$\overline{ALD}$: `0xFFFFFF04`

SBY: `0xFFFFFF08`

Allophones in Memory

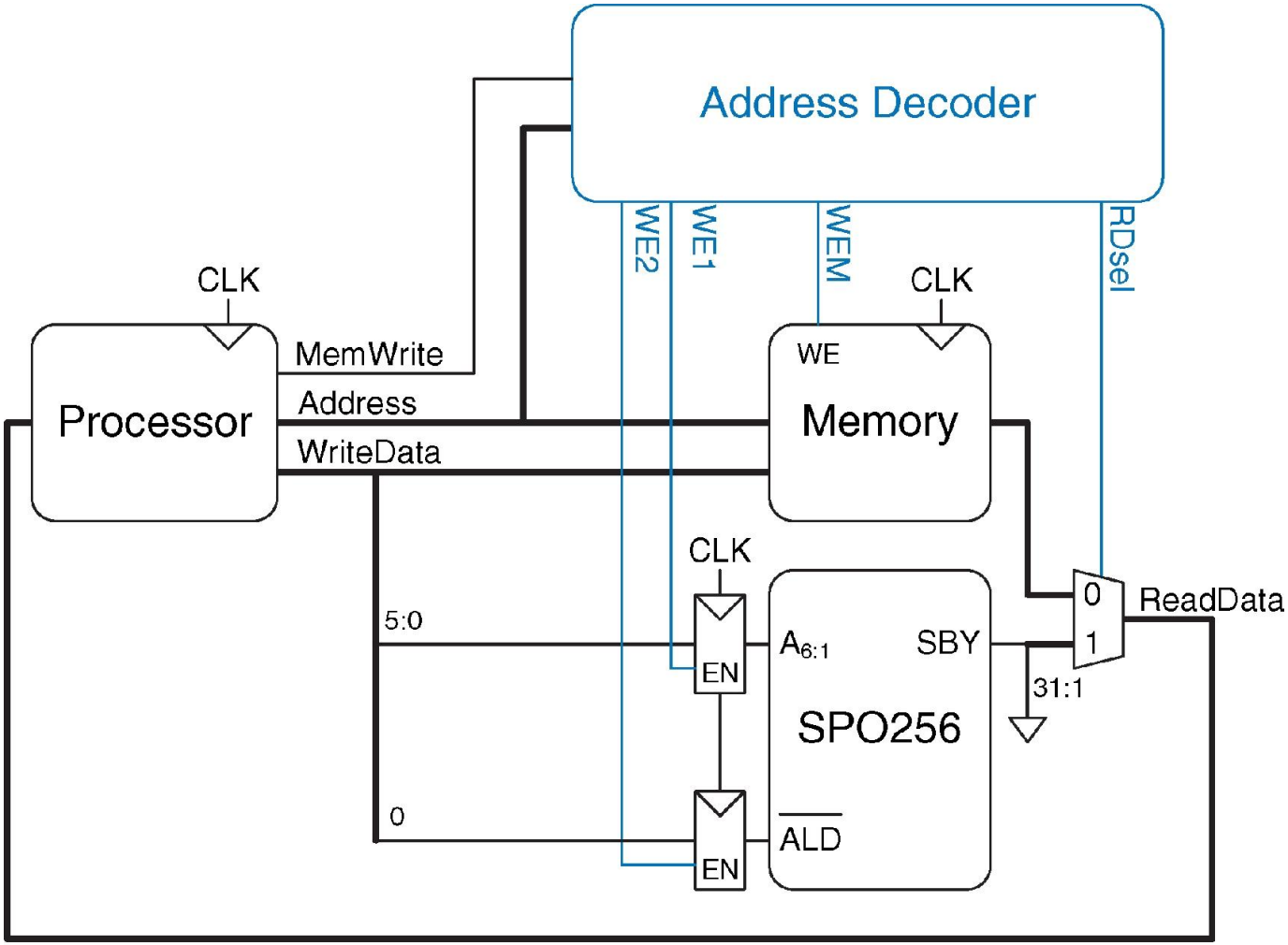
Address	Data
10000010	0x20
1000000C	0x0F
10000008	0x2D
10000004	0x07
10000000	0x1B
⋮	⋮

Main Memory

Software Driver for the Speech Chip

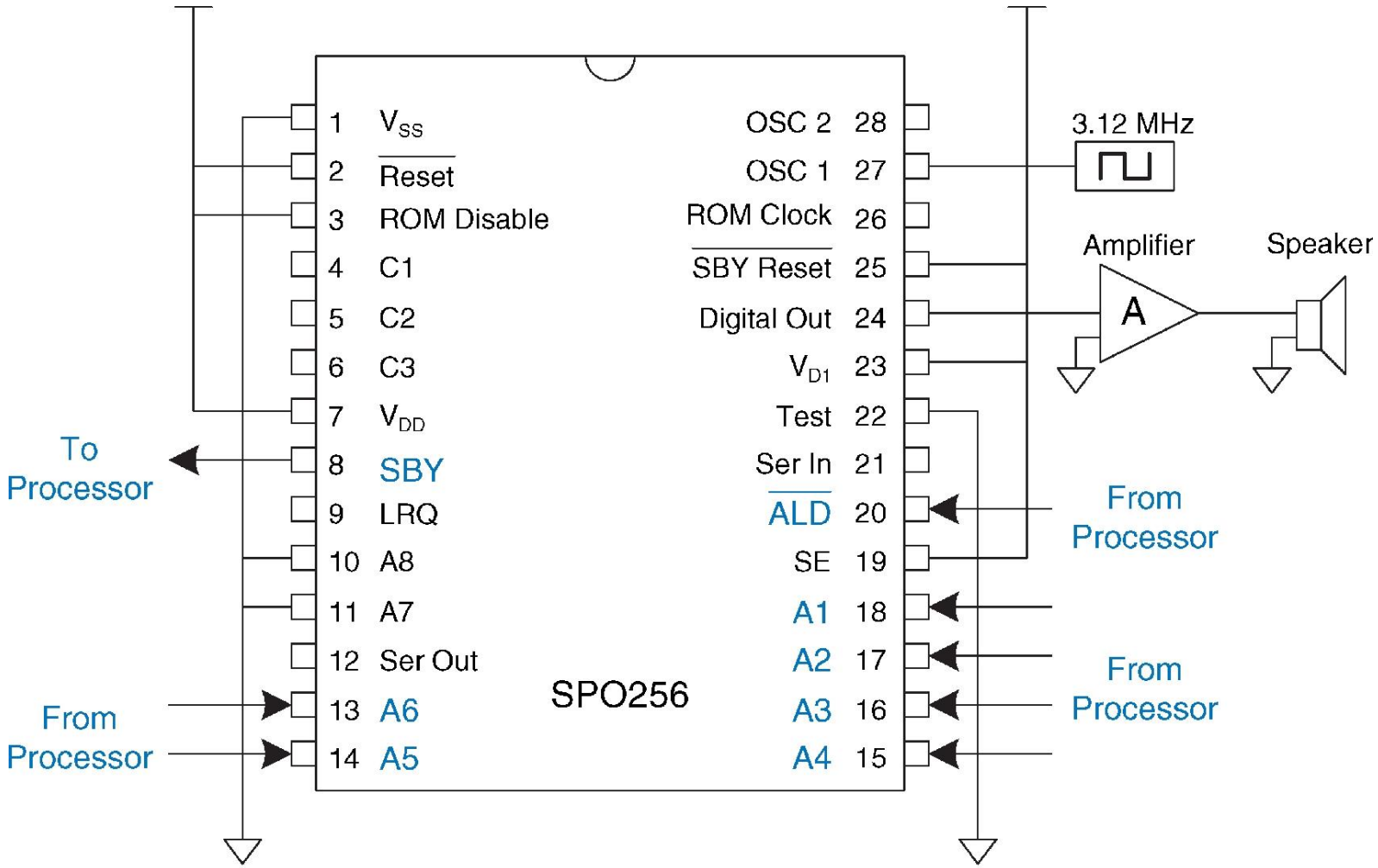
```
init:   addi $t1, $0, 1        # $t1 = 1
        addi $t2, $0, 20       # $t2 = array size * 4
        lui  $t3, 0x1000       # $t3 = array base address
        addi $t4, $0, 0        # $t4 = 0 (array index)

start:  sw   $t1, 0xFF04($0)   # ALD = 1
loop:   lw   $t5, 0xFF08($0)   # $t5 = SBY
        beq  $0,  $t5, loop    # loop until SBY == 1

        add  $t5, $t3, $t4     # $t5 = address of allophone
        lw   $t5, 0($t5)       # $t5 = allophone
        sw   $t5, 0xFF00($0)   # A6:1 = allophone
        sw   $0,  0xFF04($0)   # ALD = 0 to initiate speech
        addi $t4, $t4, 4       # increment array index
        beq  $t4, $t2, done    # last allophone in array?
        j    start             # repeat
done:
```

Hardware for Supporting SP0256



© 2007 Elsevier, Inc. All rights reserved

SP0256 Pin Connections



© 2007 Elsevier, Inc. All rights reserved

Summary

- **Processors access I/O devices just like memory**
- **A range of memory addresses is reserved for I/O**
- **An *address decoder* detects when we access I/O**
  - It enables the I/O device or memory for writing
  - Selects between the I/O device or memory for reading
- **A *device driver* is customized software routine to allow interfacing the I/O device.**
  - Device driver knows how the external hardware needs to be accessed.