processor. Section 8.6 discussed a few extensions to the programmable processor architecture such as memory-mapped I/O.

Programmable processors are typically produced in huge quantities, numbering in the tens of millions or even billions, and so tremendous attention is given to their design. Readers should realize that the programmable processor designs in this chapter are extremely simplistic and used for illustration purposes only. Yet, seeing even the simplistic designs hopefully provides an understanding of the principle of how a programmable processor works. Modern commercial processors are based on the same principles—instructions are stored as machine code in program memory, control units fetch, decode, and execute the instructions, and datapaths support the operations of the instructions using register files and ALUs. Modern processors just do a much better job, using concurrency, pipelining, and many other techniques to obtain high clock frequencies and fast program execution.

▶ 8.8 EXERCISES

SECTION 8.2: BASIC ARCHITECTURE

8.1 If a processor's program counter is 20 bits wide, up to how many words can the processor's instruction memory hold (ignoring any special tricks to expand the instruction memory size)?

8.2 Which of the following are legal single-cycle datapath operations for the datapath in Figure 8.2? Explain your answer.
(a) Copy data from a memory location into another memory location.
(b) Copy two register locations into two memory locations.
(c) Add data from a register file location and a memory location, storing the result in a memory location.

8.3 Which of the following are legal single-cycle datapath operations for the datapath in Figure 8.2? Explain your answer.
(a) Copy data from a register file location into a memory location.
(b) Subtract data from two memory locations and store the result in another memory location.
(c) Add data from a register file location and a memory location, storing the result in the same memory location.

8.4 Assume we are using a dual-port memory from which we can read two locations simultaneously. Modify the datapath of the programmable processor of Figure 8.2 to support an instruction that performs an ALU operation on any two memory locations and stores the result in a register file location. Trace through the execution of this operation, as illustrated in Figure 8.3.

8.5 Determine the operations required to instruct the datapath of Figure 8.2 to perform the operation: $D[8] = (D[4] + D[5]) - D[7]$, where D represents the data memory.

SECTION 8.3: A THREE-INSTRUCTION PROGRAMMABLE PROCESSOR

8.6 If a processor's instruction has 4 bits for the opcode, how many possible instructions can the processor support?

8.7 What does the following assembly program, which uses the three-instruction instruction set of this chapter, compute? *MOV R5, 19; ADD R5, R5, R5; MOV 20, R5.*

8.8 What does the following assembly program, which uses the three-instruction instruction set of this chapter, compute? *MOV R4, 20; MOV R9, 18; ADD R4, R4, R9; MOV R5, 30; ADD R9, R4, R5; MOV 20, R9.*

8.9 Using the three-instruction instruction set of this chapter, write an assembly program that updates the data memory D as follows: $D[0] = D[0] + D[1]$.

8.10 Using the three-instruction instruction set of this chapter, write an assembly program that updates the data memory D as follows: $D[4] = D[1]*2 + D[2]$.

8.11 Convert the following assembly program to machine code based on the three-instruction instruction set of this chapter: *MOV R5, 19; ADD R5, R5, R5; MOV 20, R5.*

8.12 List the basic register/memory transfers and operations that occur during each clock cycle for the following program, based on the three-instruction instruction set of this chapter: *MOV R0, 1; MOV R1, 9; ADD R0, R0, R1.*

SECTION 8.4: A SIX-INSTRUCTION PROGRAMMABLE PROCESSOR

8.13 List the basic register/memory transfers and operations that occur during each clock cycle for the following program, based on the six-instruction instruction set of this chapter, assuming that the content of $D[9]$ is 0: *MOV R6, #1; MOV R5, 9; JMPZ R5, label1; ADD R5, R5, R6; label1: ADD R5, R5, R6.* What is the value in *R5* after the program completes?

8.14 Add a new instruction to the six-instruction instruction set of this chapter that performs a bitwise AND of two registers and stores the result in a third register. Extend the datapath, the control unit, and the controller's FSM as needed.

8.15 Add a new instruction to the six-instruction instruction set of this chapter that performs an unconditional jump (jumps always) to a location specified by a 12-bit offset. Extend the datapath, the control unit, and the controller's FSM as needed.

8.16 Add a new instruction to the six-instruction instruction set of this chapter that performs a jump if two registers are equal, to a location specified by a 12-bit offset. Extend the datapath, the control unit, and the controller's FSM as needed.

8.17 Using the six-instruction instruction set of this chapter, write an assembly program for the C code in Figure 8.16, which computes the sum of the first N numbers, where N is another name for $D[9]$. *Hint:* Use a register to first store N.

```
i=1;
sum=0;
while (i!=N) {
  sum = sum + i;
  i = i + 1;
}
```

Figure 8.16 C code.

8.18 Using the extended instruction set you designed in Exercise 8.16, write an assembly program for the C code in Exercise 8.17.

SECTION 8.5: EXAMPLE ASSEMBLY AND MACHINE PROGRAMS

8.19 Define two new data movement instructions for the six-instruction instruction set of this chapter. Extend the datapath, the control unit, and the controller's FSM as needed.

8.20 Define two new arithmetic/logic instructions for the six-instruction instruction set of this chapter. Extend the datapath, the control unit, and the controller's FSM as needed.

engineering a
Carole has
the original
extension of
processors. "
used to com
instructions
applications
architects to
motivate the
new instructi
the benefits a
few frames pe
frames per s
everyone." A
very proud
Pentium proc
rewarding to
all of these m
everywhere."
Carole was
Hewlett-Packa

8.21 Define two new flow-of-control instructions for the six-instruction instruction set of this chapter. Extend the datapath, the control unit, and the controller's FSM as needed.

8.22 Assuming that the microprocessor's external pins *I0...I7* and *P0...P7* are mapped to data memory locations as in Figure 8.15 and an AND instruction has been added to the six-instruction instruction set of this chapter, create an assembly program that will output `0` on *P4* if all eight inputs *I0...I7* are `1`s.

▶ DESIGNER PROFILE

Carole grew up in a country where the best students went to engineering school, as engineering was highly respected. "I was good in school, so engineering seemed like a natural option. I was also very interested in building things, and very curious about how one builds new things—so I was attracted to engineering at an early age, around 10 years of age."

Carole has worked at Intel for 15 years. She was one of the original architects of the popular MMX (multimedia extension of the Intel architecture) part of Pentium processors. "It was fascinating to learn the algorithms used to compress video and audio, and to invent new instructions for the Intel Architecture to run these applications efficiently. It is not always easy for processor architects to quantify the benefits of new features, and to motivate the expense in silicon area (or chip die size) for new instructions. In the case of multimedia applications, the benefits are well understood: running a video clip at a few frames per second, or running it in real time (about 30 frames per second) makes a huge, visible difference to everyone." As is the case with so many engineers, she is very proud of what she accomplished: "When the first Pentium processor with MMX came up, it was really rewarding to think that a small piece of my mind was in all of these machines running video real time popping up everywhere."

Carole was also one of the architects on the Intel / Hewlett-Packard team that defined the Itanium computer architecture. "This was a unique opportunity to define a processor 'from scratch.' Technically this was a very challenging project, and working with so many top-notch architects was very enriching. But I also learned what it takes to build something big, involving a very large team, and two large companies. The two companies had different cultures, different methodologies, and reconciling the differences was sometimes more challenging than solving the technical problems. But this is all part of 'building things,' and this was a great lesson in leadership."

What Carole likes most about her career is "the constant change. After 22 years as a computer architect, I am still doing new things every day. Computer science is a work in progress, and it offers new opportunities that one has to grab, and run with. This is where the fun is."

Asked to give some advice to students, Carole suggests two things:

- "Stay at school as long as possible. Get a PhD if you can. To be able to adapt to constant change, you will need a very robust, and theoretical foundation. Only learning how to do things is not enough; it will get you a job for 2 years, but then your skills will be obsolete."
- "Be open for change. It is important to build an in-depth expertise in one area; in my case, it is computer architecture. But one has to be ready to use this expertise in many different projects, with different people, and more and more in different parts of the world. Fifteen years ago multimedia applications were the focus of many computer architects. Today it is bioinformatics and data mining. Change requires a lot of work to learn new domains, but not adapting to change is not an option."