How to Build a Machine

Thor Kristoffersen

September 29, 2020

This film contains programs and data. The programs can be executed by a machine to convert the data into documents, pictures, sound, and moving pictures. The purpose of this part of the film is to guide you through the process of building this machine. To understand the guide, it is necessary to have knowledge of basic 20th-century mathematics and physics.

When you have finished the machine, you will need to install an initial program in it before you can use it to execute programs. This initial program is supplied as part of the film.

You will start by building a very simple machine, and then you will extend this machine through several stages, so that by the end you will have a fully functional machine. Only when you have obtained the correct test results for each stage should you proceed with the following stage.

Number systems used here include decimal (number base 10), binary (number base 2), and hexadecimal (number base 16). All non-decimal numbers are explictly indicated by subscripts indicating the number base in decimal. Further detail on binary and hexadecimal notation can be found in Appendix A.

1 Memory

To build the machine, you first need some form of memory to represent state consisting of binary digits (bits).

1.1 Representing State

An *element* is a group of bits in the memory that forms a unit with respect to naming. The number of bits in an element is called the *size* of the element. Each element has a unique name, so that it can be unambiguously referred to. The following elements are employed by the machine:

1-bit elements T is a one 1-bit element.

8-bit elements M is an array of 2^N 8-bit elements that is indexed by an integer in the range from 0 to $2^N - 1$, for $N \leq 64$. The element at index i in M is named $M[i]$. N is a constant that must be known at the time the machine is started, and it will be given as part of the documentation of the program to be executed.

64-bit elements P and S are 64-bit elements.

In addition to the elements defined in this list, it will be necessary for some tasks to use some temporary elements. This is explained in Section 1.3

1.2 Notation

Definition 1. For any element, x, the notation $x.\mathrm{bit}[i]$ refers to bit i in x. (The numbering of bits is defined in Appendix A.1.)

Definition 2. A group of 8 bits is called an *octet*, and in a 64-bit element, x, the notation $x.\text{octet}[i]$ refers to octet i in x, which is the octet consisting of the bits from $x.\text{bit}[8i+7]$ to $x.\text{bit}[8i]$.

Definition 3. The contents of M are written as a table where the left column is the index, i, and the right column is $M[i]$. For example,

0_{16}	$\texttt{A0}_{16}$
1_{16}	$\texttt{A1}_{16}$
2_{16}	$\texttt{A2}_{16}$
3_{16}	$\texttt{A3}_{16}$

so in this case $M[2_{16}] = \texttt{A2}_{16}$.

Definition 4. The modulo operation, $x \bmod y$, for $y > 0$, is defined here as follows:

$$x \bmod y = x - y \left\lfloor \frac{x}{y} \right\rfloor$$

where $\lfloor x \rfloor$ is defined as the unique integer such that $\lfloor x \rfloor \leq x < (\lfloor x \rfloor + 1)$.

1.3 Basic Memory Operations

The following are basic memory operations that must be exist in your implementation.

Definition 5. "The value of an element" means the contents as retrieved from that element.

Definition 6. "To set an element to x" means to store the value of x in that element.

Definition 7. "The value of an octet" means the contents as retrieved from that octet.

Definition 8. "To set an octet to x" means to store the value of x in that octet, while keeping all other contents unchanged.

Definition 9. "To increment an element by n" means to set x to the value of the element and then set that element to $(x+n) \bmod 2^s$, where s is the size of the element.

Definition 10. "To decrement an element by n" means to set x to the value of the element and then set that element to $(x-n) \bmod 2^s$, where s is the size of the element.

Definition 11. "To initialize the temporary n-bit element x to k" means to create a temporary element of size n, name it x, and store the value of k in x.

1.4 Testing the Memory Operations

To test that the memory operations work as they should, do the following two tests 1000 times.

Test 1. Verify that your operations for storing and getting the value of an element work for every element, x, of size s, including P, S, T, and $M[i]$, where $i \in \{0, \ldots, 2^N - 1\}$ and $N \geq 8$.

1 Initialize the temporary elements, y and z, of size s.
2 For all $i \in \{0, \ldots, s-1\}$, set $y.\text{bit}[i]$ to a random bit value.
3 For all $i \in \{0, \ldots, s-1\}$, set $z.\text{bit}[i]$ to a random bit value.
4 Set x to y.
5 Verify that $x = y$.

6 Increment x by z.
7 Verify that $x = (y+z) \bmod 2^s$.
8 Decrement x by z.
9 Verify that $x = y$.

Test 2. Verify that your octet operations work for every 64-bit element, x, and every octet, $x.\text{octet}[n]$ where $n \in \{0, \ldots, 7\}$.
1 Initialize the temporary element, y, of size 64.
2 Initialize the temporary element, z, of size 8.
3 For all $i \in \{0, \ldots, s-1\}$, set $y.\text{bit}[i]$ to a random bit value.
4 For all $i \in \{0, \ldots, 7\}$, set $z.\text{bit}[i]$ to a random bit value.
5 Set x to y.
6 Set $x.\text{octet}[n]$ to z.
7 Verify that $x.\text{octet}[n] = z$.
8 Verify that $x.\text{octet}[i] = y.\text{octet}[i]$ where $i \in \{0, \ldots, 7\}$ and $i \neq n$.

1.5 Summary of Memory

The entire state of the machine is contained in elements P, S, T, M, and the constant N.

2 Basic Procedures

You will need to implement six basic procedures that are frequently used by the machine. These use the memory operations described in Section 1.3.

2.1 PUT Octets

Assume that we have two temporary 64-bit elements called x and a, and that n is either 1, 2, 4, or 8. The expression "PUT n octets from x into index a" means to do the following steps:
1 Initialize the temporary 8-bit element i to 0.
2 Do the following n times:
  a Set $M[a+i]$ to $x.\text{octet}[i]$.
  b Increment i by 1.

Test 3. To test PUT, do as follows, with $N = 4$:
1 Set $M[i]$ to 0 for all $i \in \{0, \ldots, 2^N - 1\}$.
2 Initialize the temporary 64-bit element x to $\texttt{A7A6A5A4A3A2A1A0}_{16}$.
3 PUT 1 octet from x into index $\texttt{01}_{16}$.
4 PUT 2 octets from x into index $\texttt{02}_{16}$.
5 PUT 4 octets from x into index $\texttt{04}_{16}$.
6 PUT 8 octets from x into index $\texttt{08}_{16}$.
7 Verify that the contents of the elements in M are as in the following table.

$\texttt{0}_{16}$	$\texttt{00}_{16}$	$\texttt{4}_{16}$	$\texttt{A0}_{16}$	$\texttt{8}_{16}$	$\texttt{A0}_{16}$	$\texttt{C}_{16}$	$\texttt{A4}_{16}$
$\texttt{1}_{16}$	$\texttt{A0}_{16}$	$\texttt{5}_{16}$	$\texttt{A1}_{16}$	$\texttt{9}_{16}$	$\texttt{A1}_{16}$	$\texttt{D}_{16}$	$\texttt{A5}_{16}$
$\texttt{2}_{16}$	$\texttt{A0}_{16}$	$\texttt{6}_{16}$	$\texttt{A2}_{16}$	$\texttt{A}_{16}$	$\texttt{A2}_{16}$	$\texttt{E}_{16}$	$\texttt{A6}_{16}$
$\texttt{3}_{16}$	$\texttt{A1}_{16}$	$\texttt{7}_{16}$	$\texttt{A3}_{16}$	$\texttt{B}_{16}$	$\texttt{A3}_{16}$	$\texttt{F}_{16}$	$\texttt{A7}_{16}$

2.2 GET Octets

Assume that we have a temporary 64-bit element called a and that n is either 1, 2, 4, or 8. The expression "GET n octets from index a into x" means to do the following steps:

1 Initialize the temporary 64-bit element x to 0.
2 Initialize the temporary 8-bit element i to 0.
3 Do the following n times:
  - **a** Set x.octet$[i]$ to $M[a+i]$.
  - **b** Increment i by 1.

The temporary 64-bit element x can now be used by other procedures.

Test 4. To test GET, first set $N = 4$, and set the following elements of M to the provided values.

$\texttt{0}_{16}$	$\texttt{A0}_{16}$	$\texttt{4}_{16}$	$\texttt{A4}_{16}$	$\texttt{8}_{16}$	$\texttt{A8}_{16}$	$\texttt{C}_{16}$	$\texttt{AC}_{16}$
$\texttt{1}_{16}$	$\texttt{A1}_{16}$	$\texttt{5}_{16}$	$\texttt{A5}_{16}$	$\texttt{9}_{16}$	$\texttt{A9}_{16}$	$\texttt{D}_{16}$	$\texttt{AD}_{16}$
$\texttt{2}_{16}$	$\texttt{A2}_{16}$	$\texttt{6}_{16}$	$\texttt{A6}_{16}$	$\texttt{A}_{16}$	$\texttt{AA}_{16}$	$\texttt{E}_{16}$	$\texttt{AE}_{16}$
$\texttt{3}_{16}$	$\texttt{A3}_{16}$	$\texttt{7}_{16}$	$\texttt{A7}_{16}$	$\texttt{B}_{16}$	$\texttt{AB}_{16}$	$\texttt{F}_{16}$	$\texttt{AF}_{16}$

Then proceed as follows:

1 Initialize the temporary 64-bit element x to 0.
2 GET 1 octet from index $\texttt{01}_{16}$ into x.
3 Verify that the value of x is $\texttt{00000000000000A1}_{16}$.
4 GET 2 octets from index $\texttt{02}_{16}$ into x.
5 Verify that the value of x is $\texttt{000000000000A3A2}_{16}$.
6 GET 4 octets from index $\texttt{04}_{16}$ into x.
7 Verify that the value of x is $\texttt{00000000A7A6A5A4}_{16}$.
8 GET 8 octets from index $\texttt{08}_{16}$ into x.
9 Verify that the value of x is $\texttt{AFAEADACABAAA9A8}_{16}$.

2.3 PUSH Element

Assume that we have a temporary 64-bit element called x. The expression "PUSH x" means to do the following steps:

1 Decrement S by 8.
2 PUT 8 octets from x into index S.

Test 5. To test PUSH, do as follows, with $N = 4$:

1 Set $M[i]$ to 0 for all $i \in \{0, \ldots, 2^N - 1\}$.
2 Set S to $\texttt{10}_{16}$.
3 Initialize the temporary 64-bit element x to $\texttt{AFAEADACABAAA9A8}_{16}$.
4 Initialize the temporary 64-bit element y to $\texttt{A7A6A5A4A3A2A1A0}_{16}$.
5 PUSH x.
6 Verify that the value of S is $\texttt{08}_{16}$.
7 PUSH y.
8 Verify that the value of S is $\texttt{00}_{16}$.
9 Verify that the contents of the elements in M are as in the following table.

$\texttt{0}_{16}$	$\texttt{A0}_{16}$	$\texttt{4}_{16}$	$\texttt{A4}_{16}$	$\texttt{8}_{16}$	$\texttt{A8}_{16}$	$\texttt{C}_{16}$	$\texttt{AC}_{16}$
$\texttt{1}_{16}$	$\texttt{A1}_{16}$	$\texttt{5}_{16}$	$\texttt{A5}_{16}$	$\texttt{9}_{16}$	$\texttt{A9}_{16}$	$\texttt{D}_{16}$	$\texttt{AD}_{16}$
$\texttt{2}_{16}$	$\texttt{A2}_{16}$	$\texttt{6}_{16}$	$\texttt{A6}_{16}$	$\texttt{A}_{16}$	$\texttt{AA}_{16}$	$\texttt{E}_{16}$	$\texttt{AE}_{16}$
$\texttt{3}_{16}$	$\texttt{A3}_{16}$	$\texttt{7}_{16}$	$\texttt{A7}_{16}$	$\texttt{B}_{16}$	$\texttt{AB}_{16}$	$\texttt{F}_{16}$	$\texttt{AF}_{16}$

2.4 POP Element

The expression "POP x" means to do the following steps:

1 Initialize the temporary 64-bit element x to 0.
2 GET 8 octets from index S into x.
3 Increment S by 8.

The 64-bit element x can now be used by other procedures.

Test 6. To test POP, first set $N = 4$, set $S = 0$, and set the following elements of M to the provided values.

$\mathtt{0}_{16}$	$\mathtt{A0}_{16}$	$\mathtt{4}_{16}$	$\mathtt{A4}_{16}$	$\mathtt{8}_{16}$	$\mathtt{A8}_{16}$	$\mathtt{C}_{16}$	$\mathtt{AC}_{16}$
$\mathtt{1}_{16}$	$\mathtt{A1}_{16}$	$\mathtt{5}_{16}$	$\mathtt{A5}_{16}$	$\mathtt{9}_{16}$	$\mathtt{A9}_{16}$	$\mathtt{D}_{16}$	$\mathtt{AD}_{16}$
$\mathtt{2}_{16}$	$\mathtt{A2}_{16}$	$\mathtt{6}_{16}$	$\mathtt{A6}_{16}$	$\mathtt{A}_{16}$	$\mathtt{AA}_{16}$	$\mathtt{E}_{16}$	$\mathtt{AE}_{16}$
$\mathtt{3}_{16}$	$\mathtt{A3}_{16}$	$\mathtt{7}_{16}$	$\mathtt{A7}_{16}$	$\mathtt{B}_{16}$	$\mathtt{AB}_{16}$	$\mathtt{F}_{16}$	$\mathtt{AF}_{16}$

Then proceed as follows:

1 Initialize the temporary 64-bit element x to 0.
2 Initialize the temporary 64-bit element y to 0.
3 POP x.
4 Verify that $S = \mathtt{08}_{16}$ and that $x = \mathtt{A7A6A5A4A3A2A1A0}_{16}$.
5 POP y.
6 Verify that $S = \mathtt{10}_{16}$ and that $y = \mathtt{AFAEADACABAAA9A8}_{16}$.

2.5 FETCH Octets

Assume that n is either 1, 2, 4, or 8. The expression "FETCH n octets into x" means to do the following steps:

1 Initialize the temporary 64-bit element x to 0.
2 GET n octets from index P into x.
3 Increment P by n.

The temporary 64-bit element x can now be used by other procedures.

Test 7. To test FETCH, first set $N = 4$, and set the following elements of M to the provided values.

$\mathtt{0}_{16}$	$\mathtt{A0}_{16}$	$\mathtt{4}_{16}$	$\mathtt{A4}_{16}$	$\mathtt{8}_{16}$	$\mathtt{A8}_{16}$	$\mathtt{C}_{16}$	$\mathtt{AC}_{16}$
$\mathtt{1}_{16}$	$\mathtt{A1}_{16}$	$\mathtt{5}_{16}$	$\mathtt{A5}_{16}$	$\mathtt{9}_{16}$	$\mathtt{A9}_{16}$	$\mathtt{D}_{16}$	$\mathtt{AD}_{16}$
$\mathtt{2}_{16}$	$\mathtt{A2}_{16}$	$\mathtt{6}_{16}$	$\mathtt{A6}_{16}$	$\mathtt{A}_{16}$	$\mathtt{AA}_{16}$	$\mathtt{E}_{16}$	$\mathtt{AE}_{16}$
$\mathtt{3}_{16}$	$\mathtt{A3}_{16}$	$\mathtt{7}_{16}$	$\mathtt{A7}_{16}$	$\mathtt{B}_{16}$	$\mathtt{AB}_{16}$	$\mathtt{F}_{16}$	$\mathtt{AF}_{16}$

Then proceed as follows:

1 Initialize the temporary 64-bit element w to 0.
2 FETCH 1 octet into w.
3 Verify that $P = \mathtt{01}_{16}$ and that $w = \mathtt{00000000000000A0}_{16}$.
4 FETCH 2 octets into w.
5 Verify that $P = \mathtt{03}_{16}$ and that $w = \mathtt{000000000000A2A1}_{16}$.
6 FETCH 4 octets into w.
7 Verify that $P = \mathtt{07}_{16}$ and that $z = \mathtt{00000000A6A5A4A3}_{16}$.
8 FETCH 8 octets into w.
9 Verify that $P = \mathtt{0F}_{16}$ and that $w = \mathtt{AEADACABAAA9A8A7}_{16}$.

2.6 EXTEND Octets

Assume that we have a temporary 64-bit element called x and that n is either 1, 2, or 4. The expression "EXTEND n octets in x" means to do the following step:

1 If the value of $x.\mathrm{bit}[8n-1]$ is 1, then for all $i \in \{8n,\ldots,63\}$ set $x.\mathrm{bit}[i]$ to 1.

The temporary 64-bit element x can now be used by other procedures.

Test 8. To test EXTEND, do as follows:

1 Test EXTEND 1 octet.
- **1** Initialize the temporary 64-bit element x to $\texttt{000000000000007F}_{16}$.
- **2** EXTEND 1 octet in x.
- **3** Verify that the value of x is $\texttt{000000000000007F}_{16}$.
- **4** Increment x by 1.
- **5** EXTEND 1 octet in x.
- **6** Verify that the value of x is $\texttt{FFFFFFFFFFFFFF80}_{16}$.

2 Test EXTEND 2 octets.
- **1** Initialize the temporary 64-bit element y to $\texttt{0000000000007FFF}_{16}$.
- **2** EXTEND 2 octets in y.
- **3** Verify that the value of y is $\texttt{0000000000007FFF}_{16}$.
- **4** Increment y by 1.
- **5** EXTEND 2 octets in y.
- **6** Verify that the value of y is $\texttt{FFFFFFFFFFFF8000}_{16}$.

3 Test EXTEND 4 octets.
- **1** Initialize the temporary 64-bit element z to $\texttt{000000007FFFFFFF}_{16}$.
- **2** EXTEND 4 octets in z.
- **3** Verify that the value of z is $\texttt{000000007FFFFFFF}_{16}$.
- **4** Increment z by 1.
- **5** EXTEND 4 octets in z.
- **6** Verify that the value of z is $\texttt{FFFFFFFF80000000}_{16}$.

3 A Very Basic Machine

You can now proceed with implementing a basic machine with $N = 8$.

Before the machine is started, a program must be stored in M. The program is a sequence of n octets, which are stored in $M[i]$ for $i \in \{0,\ldots,n-1\}$. The rest of M must be set to 0, that is, $M[i]$ for $i \in \{n,\ldots,2^N-1\}$.

Every time the machine starts, set T to 0, set P to 0, and set S to 2^N, then execute the main procedure repeatedly until the value of T is $\texttt{1}_2$. The main procedure must carry out the following steps:

1 Initialize the temporary 64-bit element k to 0.

2 FETCH 1 octet into k.

3 If the value of k is

$\texttt{01}_{16}$ then do nothing.

$\texttt{02}_{16}$ then
- **1** Initialize the temporary 64-bit element a to 0.
- **2** POP a.
- **3** Set P to a.

$\texttt{03}_{16}$ then

1 Initialize the temporary 64-bit elements a and x to 0.
2 Fetch 1 octet into a.
3 Extend 1 octet in a.
4 Pop x.
5 If the value of x is 0, then increment P by a.

$\texttt{04}_{16}$ then
1 Initialize the temporary 64-bit element a to 0.
2 Pop a.
3 Set S to a.

$\texttt{05}_{16}$ then
1 Initialize the temporary 64-bit element a to P.
2 Push a.

$\texttt{06}_{16}$ then
1 Initialize the temporary 64-bit element a to S.
2 Push a.

$\texttt{08}_{16}$ then
1 Initialize the temporary 64-bit element a to 0.
2 Fetch 1 octet into a.
3 Push a.

4 If the value of k does not occur in the list in the previous point, then set T to $\texttt{1}_2$.

Test 9. To test the machine, set the following elements of M to the provided values and start the machine.

$\texttt{00}_{16}$	$\texttt{01}_{16}$	$\texttt{04}_{16}$	$\texttt{01}_{16}$	$\texttt{08}_{16}$	$\texttt{01}_{16}$	$\texttt{0C}_{16}$	$\texttt{02}_{16}$	$\texttt{10}_{16}$	$\texttt{00}_{16}$	$\texttt{14}_{16}$	$\texttt{08}_{16}$
$\texttt{01}_{16}$	$\texttt{08}_{16}$	$\texttt{05}_{16}$	$\texttt{08}_{16}$	$\texttt{09}_{16}$	$\texttt{00}_{16}$	$\texttt{0D}_{16}$	$\texttt{00}_{16}$	$\texttt{11}_{16}$	$\texttt{00}_{16}$	$\texttt{15}_{16}$	$\texttt{F8}_{16}$
$\texttt{02}_{16}$	$\texttt{12}_{16}$	$\texttt{06}_{16}$	$\texttt{00}_{16}$	$\texttt{0A}_{16}$	$\texttt{03}_{16}$	$\texttt{0E}_{16}$	$\texttt{00}_{16}$	$\texttt{12}_{16}$	$\texttt{06}_{16}$	$\texttt{16}_{16}$	$\texttt{04}_{16}$
$\texttt{03}_{16}$	$\texttt{08}_{16}$	$\texttt{07}_{16}$	$\texttt{03}_{16}$	$\texttt{0B}_{16}$	$\texttt{01}_{16}$	$\texttt{0F}_{16}$	$\texttt{00}_{16}$	$\texttt{13}_{16}$	$\texttt{05}_{16}$	$\texttt{17}_{16}$	$\texttt{00}_{16}$

When the machine terminates, the value of T should be $\texttt{1}_2$, the value of P should be $\texttt{18}_{16}$, the value of S should be $\texttt{F8}_{16}$, and the following elements of M should have the indicated values.

$\texttt{E8}_{16}$	$\texttt{F8}_{16}$	$\texttt{EC}_{16}$	$\texttt{00}_{16}$	$\texttt{F0}_{16}$	$\texttt{14}_{16}$	$\texttt{F4}_{16}$	$\texttt{00}_{16}$	$\texttt{F8}_{16}$	$\texttt{00}_{16}$	$\texttt{FC}_{16}$	$\texttt{00}_{16}$
$\texttt{E9}_{16}$	$\texttt{00}_{16}$	$\texttt{ED}_{16}$	$\texttt{00}_{16}$	$\texttt{F1}_{16}$	$\texttt{00}_{16}$	$\texttt{F5}_{16}$	$\texttt{00}_{16}$	$\texttt{F9}_{16}$	$\texttt{01}_{16}$	$\texttt{FD}_{16}$	$\texttt{00}_{16}$
$\texttt{EA}_{16}$	$\texttt{00}_{16}$	$\texttt{EE}_{16}$	$\texttt{00}_{16}$	$\texttt{F2}_{16}$	$\texttt{00}_{16}$	$\texttt{F6}_{16}$	$\texttt{00}_{16}$	$\texttt{FA}_{16}$	$\texttt{00}_{16}$	$\texttt{FE}_{16}$	$\texttt{00}_{16}$
$\texttt{EB}_{16}$	$\texttt{00}_{16}$	$\texttt{EF}_{16}$	$\texttt{00}_{16}$	$\texttt{F3}_{16}$	$\texttt{00}_{16}$	$\texttt{F7}_{16}$	$\texttt{00}_{16}$	$\texttt{FB}_{16}$	$\texttt{00}_{16}$	$\texttt{FF}_{16}$	$\texttt{00}_{16}$

All other $M[i]$ elements should remain unchanged.

4 Adding Bit-Copying Capabilities

You can now add several more cases to step **3** of the main procedure. These cases add bit-copying capabilities to the machine.

3 If k is

$\texttt{10}_{16}$ then
1 Initialize the temporary 64-bit elements a and x to 0.
2 Pop a.
3 Get 1 octet from index a into x.
4 Push x.

$\mathtt{11}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Get 2 octets from index a into x.
4. Push x.

$\mathtt{12}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Get 4 octets from index a into x.
4. Push x.

$\mathtt{13}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Get 8 octets from index a into x.
4. Push x.

$\mathtt{14}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Pop x.
4. Put 1 octet from x into index a.

$\mathtt{15}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Pop x.
4. Put 2 octets from x into index a.

$\mathtt{16}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Pop x.
4. Put 4 octets from x into index a.

$\mathtt{17}_{16}$ then
1. Initialize the temporary 64-bit elements a and x to 0.
2. Pop a.
3. Pop x.
4. Put 8 octets from x into index a.

Test 10. To test the machine, set the following elements of M to the provided values and start the machine.

$\mathtt{00}_{16}$	$\mathtt{08}_{16}$	$\mathtt{08}_{16}$	$\mathtt{12}_{16}$	$\mathtt{10}_{16}$	$\mathtt{EC}_{16}$	$\mathtt{18}_{16}$	$\mathtt{00}_{16}$	$\mathtt{20}_{16}$	$\mathtt{A7}_{16}$
$\mathtt{01}_{16}$	$\mathtt{19}_{16}$	$\mathtt{09}_{16}$	$\mathtt{08}_{16}$	$\mathtt{11}_{16}$	$\mathtt{15}_{16}$	$\mathtt{19}_{16}$	$\mathtt{A0}_{16}$	$\mathtt{21}_{16}$	$\mathtt{A8}_{16}$
$\mathtt{02}_{16}$	$\mathtt{13}_{16}$	$\mathtt{0A}_{16}$	$\mathtt{E8}_{16}$	$\mathtt{12}_{16}$	$\mathtt{08}_{16}$	$\mathtt{1A}_{16}$	$\mathtt{A1}_{16}$	$\mathtt{22}_{16}$	$\mathtt{A9}_{16}$
$\mathtt{03}_{16}$	$\mathtt{08}_{16}$	$\mathtt{0B}_{16}$	$\mathtt{16}_{16}$	$\mathtt{13}_{16}$	$\mathtt{27}_{16}$	$\mathtt{1B}_{16}$	$\mathtt{A2}_{16}$	$\mathtt{23}_{16}$	$\mathtt{AA}_{16}$
$\mathtt{04}_{16}$	$\mathtt{E0}_{16}$	$\mathtt{0C}_{16}$	$\mathtt{08}_{16}$	$\mathtt{14}_{16}$	$\mathtt{10}_{16}$	$\mathtt{1C}_{16}$	$\mathtt{A3}_{16}$	$\mathtt{24}_{16}$	$\mathtt{AB}_{16}$
$\mathtt{05}_{16}$	$\mathtt{17}_{16}$	$\mathtt{0D}_{16}$	$\mathtt{25}_{16}$	$\mathtt{15}_{16}$	$\mathtt{08}_{16}$	$\mathtt{1D}_{16}$	$\mathtt{A4}_{16}$	$\mathtt{25}_{16}$	$\mathtt{AC}_{16}$
$\mathtt{06}_{16}$	$\mathtt{08}_{16}$	$\mathtt{0E}_{16}$	$\mathtt{11}_{16}$	$\mathtt{16}_{16}$	$\mathtt{EE}_{16}$	$\mathtt{1E}_{16}$	$\mathtt{A5}_{16}$	$\mathtt{26}_{16}$	$\mathtt{AD}_{16}$
$\mathtt{07}_{16}$	$\mathtt{21}_{16}$	$\mathtt{0F}_{16}$	$\mathtt{08}_{16}$	$\mathtt{17}_{16}$	$\mathtt{14}_{16}$	$\mathtt{1F}_{16}$	$\mathtt{A6}_{16}$	$\mathtt{27}_{16}$	$\mathtt{AE}_{16}$

When the machine terminates, the value of S should be $\texttt{100}_{16}$, and the following elements of M should have the indicated values.

$\texttt{E0}_{16}$	$\texttt{A0}_{16}$	$\texttt{E4}_{16}$	$\texttt{A4}_{16}$	$\texttt{E8}_{16}$	$\texttt{A8}_{16}$	$\texttt{EC}_{16}$	$\texttt{AC}_{16}$
$\texttt{E1}_{16}$	$\texttt{A1}_{16}$	$\texttt{E5}_{16}$	$\texttt{A5}_{16}$	$\texttt{E9}_{16}$	$\texttt{A9}_{16}$	$\texttt{ED}_{16}$	$\texttt{AD}_{16}$
$\texttt{E2}_{16}$	$\texttt{A2}_{16}$	$\texttt{E6}_{16}$	$\texttt{A6}_{16}$	$\texttt{EA}_{16}$	$\texttt{AA}_{16}$	$\texttt{EE}_{16}$	$\texttt{AE}_{16}$
$\texttt{E3}_{16}$	$\texttt{A3}_{16}$	$\texttt{E7}_{16}$	$\texttt{A7}_{16}$	$\texttt{EB}_{16}$	$\texttt{AB}_{16}$	$\texttt{EF}_{16}$	$\texttt{00}_{16}$

5 Adding More Bit-Copying Capabilities

You can now add several more cases to step **3** of the main procedure. These cases add still more bit-copying capabilities to the machine.

3 If k is

$\texttt{07}_{16}$ then
- **1** Initialize the temporary 64-bit element a to 0.
- **2** PUSH a.

$\texttt{09}_{16}$ then
- **1** Initialize the temporary 64-bit element a to 0.
- **2** FETCH 2 octets into a.
- **3** PUSH a.

$\texttt{0A}_{16}$ then
- **1** Initialize the temporary 64-bit element a to 0.
- **2** FETCH 4 octets into a.
- **3** PUSH a.

$\texttt{0B}_{16}$ then
- **1** Initialize the temporary 64-bit element a to 0.
- **2** FETCH 8 octets into a.
- **3** PUSH a.

$\texttt{0C}_{16}$ then
- **1** Initialize the temporary 64-bit element x to 0.
- **2** POP x.
- **3** EXTEND 1 octet in x.
- **4** PUSH x.

$\texttt{0D}_{16}$ then
- **1** Initialize the temporary 64-bit element x to 0.
- **2** POP x.
- **3** EXTEND 2 octets in x.
- **4** PUSH x.

$\texttt{0E}_{16}$ then
- **1** Initialize the temporary 64-bit element x to 0.
- **2** POP x.
- **3** EXTEND 4 octets in x.
- **4** PUSH x.

Test 11. To test the machine, set the following elements of M to the provided values and start the machine.

$\texttt{00}_{16}$	$\texttt{07}_{16}$	$\texttt{04}_{16}$	$\texttt{54}_{16}$	$\texttt{08}_{16}$	$\texttt{DC}_{16}$	$\texttt{0C}_{16}$	$\texttt{BA}_{16}$	$\texttt{10}_{16}$	$\texttt{09}_{16}$	$\texttt{14}_{16}$	$\texttt{08}_{16}$
$\texttt{01}_{16}$	$\texttt{0B}_{16}$	$\texttt{05}_{16}$	$\texttt{76}_{16}$	$\texttt{09}_{16}$	$\texttt{FE}_{16}$	$\texttt{0D}_{16}$	$\texttt{DC}_{16}$	$\texttt{11}_{16}$	$\texttt{DC}_{16}$	$\texttt{15}_{16}$	$\texttt{FE}_{16}$
$\texttt{02}_{16}$	$\texttt{10}_{16}$	$\texttt{06}_{16}$	$\texttt{98}_{16}$	$\texttt{0A}_{16}$	$\texttt{0A}_{16}$	$\texttt{0E}_{16}$	$\texttt{FE}_{16}$	$\texttt{12}_{16}$	$\texttt{FE}_{16}$	$\texttt{16}_{16}$	$\texttt{0C}_{16}$
$\texttt{03}_{16}$	$\texttt{32}_{16}$	$\texttt{07}_{16}$	$\texttt{BA}_{16}$	$\texttt{0B}_{16}$	$\texttt{98}_{16}$	$\texttt{0F}_{16}$	$\texttt{0E}_{16}$	$\texttt{13}_{16}$	$\texttt{0D}_{16}$	$\texttt{17}_{16}$	$\texttt{00}_{16}$

When the machine terminates, the value of S should be $\texttt{E0}_{16}$, and the following elements of M should have the indicated values.

$\texttt{D8}_{16}$	$\texttt{FE}_{16}$	$\texttt{E0}_{16}$	$\texttt{DC}_{16}$	$\texttt{E8}_{16}$	$\texttt{98}_{16}$	$\texttt{F0}_{16}$	$\texttt{10}_{16}$	$\texttt{F8}_{16}$	$\texttt{00}_{16}$
$\texttt{D9}_{16}$	$\texttt{FF}_{16}$	$\texttt{E1}_{16}$	$\texttt{FE}_{16}$	$\texttt{E9}_{16}$	$\texttt{BA}_{16}$	$\texttt{F1}_{16}$	$\texttt{32}_{16}$	$\texttt{F9}_{16}$	$\texttt{00}_{16}$
$\texttt{DA}_{16}$	$\texttt{FF}_{16}$	$\texttt{E2}_{16}$	$\texttt{FF}_{16}$	$\texttt{EA}_{16}$	$\texttt{DC}_{16}$	$\texttt{F2}_{16}$	$\texttt{54}_{16}$	$\texttt{FA}_{16}$	$\texttt{00}_{16}$
$\texttt{DB}_{16}$	$\texttt{FF}_{16}$	$\texttt{E3}_{16}$	$\texttt{FF}_{16}$	$\texttt{EB}_{16}$	$\texttt{FE}_{16}$	$\texttt{F3}_{16}$	$\texttt{76}_{16}$	$\texttt{FB}_{16}$	$\texttt{00}_{16}$
$\texttt{DC}_{16}$	$\texttt{FF}_{16}$	$\texttt{E4}_{16}$	$\texttt{FF}_{16}$	$\texttt{EC}_{16}$	$\texttt{FF}_{16}$	$\texttt{F4}_{16}$	$\texttt{98}_{16}$	$\texttt{FC}_{16}$	$\texttt{00}_{16}$
$\texttt{DD}_{16}$	$\texttt{FF}_{16}$	$\texttt{E5}_{16}$	$\texttt{FF}_{16}$	$\texttt{ED}_{16}$	$\texttt{FF}_{16}$	$\texttt{F5}_{16}$	$\texttt{BA}_{16}$	$\texttt{FD}_{16}$	$\texttt{00}_{16}$
$\texttt{DE}_{16}$	$\texttt{FF}_{16}$	$\texttt{E6}_{16}$	$\texttt{FF}_{16}$	$\texttt{EE}_{16}$	$\texttt{FF}_{16}$	$\texttt{F6}_{16}$	$\texttt{DC}_{16}$	$\texttt{FE}_{16}$	$\texttt{00}_{16}$
$\texttt{DF}_{16}$	$\texttt{FF}_{16}$	$\texttt{E7}_{16}$	$\texttt{FF}_{16}$	$\texttt{EF}_{16}$	$\texttt{FF}_{16}$	$\texttt{F7}_{16}$	$\texttt{FE}_{16}$	$\texttt{FF}_{16}$	$\texttt{00}_{16}$

6 Adding Arithmetic

You can now add several more cases to step **3** of the main procedure. These cases add arithmetic capabilities to the machine.

3 If k is

20_{16} then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP y.
- **3** POP x.
- **4** PUSH $(x+y) \bmod 2^{64}$.

21_{16} then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP y.
- **3** POP x.
- **4** PUSH $(xy) \bmod 2^{64}$.

22_{16} then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP y.
- **3** POP x.
- **4** If $x > 0$ and $y > 0$,
  then PUSH q, such that $x = qy + r$ and $0 \leq r < y$,
  otherwise PUSH 0.

23_{16} then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP y.
- **3** POP x.
- **4** If $x > 0$ and $y > 0$,
  then PUSH r, such that $x = qy + r$ and $0 \leq r < y$,
  otherwise PUSH 0.

Test 12. To test the machine, set the following elements of M to the provided values and start the machine.

00$_{16}$	08$_{16}$	08$_{16}$	1D$_{16}$	10$_{16}$	13$_{16}$	18$_{16}$	08$_{16}$	20$_{16}$	58$_{16}$	28$_{16}$	5C$_{16}$
01$_{16}$	1D$_{16}$	09$_{16}$	13$_{16}$	11$_{16}$	08$_{16}$	19$_{16}$	25$_{16}$	21$_{16}$	1B$_{16}$	29$_{16}$	7D$_{16}$
02$_{16}$	13$_{16}$	0A$_{16}$	08$_{16}$	12$_{16}$	25$_{16}$	1A$_{16}$	13$_{16}$	22$_{16}$	C9$_{16}$	2A$_{16}$	2C$_{16}$
03$_{16}$	08$_{16}$	0B$_{16}$	25$_{16}$	13$_{16}$	13$_{16}$	1B$_{16}$	23$_{16}$	23$_{16}$	77$_{16}$	2B$_{16}$	17$_{16}$
04$_{16}$	25$_{16}$	0C$_{16}$	13$_{16}$	14$_{16}$	22$_{16}$	1C$_{16}$	00$_{16}$	24$_{16}$	FF$_{16}$	2C$_{16}$	3F$_{16}$
05$_{16}$	13$_{16}$	0D$_{16}$	21$_{16}$	15$_{16}$	08$_{16}$	1D$_{16}$	98$_{16}$	25$_{16}$	88$_{16}$		
06$_{16}$	20$_{16}$	0E$_{16}$	08$_{16}$	16$_{16}$	1D$_{16}$	1E$_{16}$	E7$_{16}$	26$_{16}$	60$_{16}$		
07$_{16}$	08$_{16}$	0F$_{16}$	1D$_{16}$	17$_{16}$	13$_{16}$	1F$_{16}$	D9$_{16}$	27$_{16}$	09$_{16}$		

When the machine terminates, the value of S should be E0$_{16}$, and the following elements of M should have the indicated values.

E0$_{16}$	78$_{16}$	E8$_{16}$	04$_{16}$	F0$_{16}$	C0$_{16}$	F8$_{16}$	20$_{16}$
E1$_{16}$	65$_{16}$	E9$_{16}$	00$_{16}$	F1$_{16}$	08$_{16}$	F9$_{16}$	48$_{16}$
E2$_{16}$	B4$_{16}$	EA$_{16}$	00$_{16}$	F2$_{16}$	F4$_{16}$	FA$_{16}$	E3$_{16}$
E3$_{16}$	E8$_{16}$	EB$_{16}$	00$_{16}$	F3$_{16}$	AE$_{16}$	FB$_{16}$	B4$_{16}$
E4$_{16}$	25$_{16}$	EC$_{16}$	00$_{16}$	F4$_{16}$	F4$_{16}$	FC$_{16}$	98$_{16}$
E5$_{16}$	17$_{16}$	ED$_{16}$	00$_{16}$	F5$_{16}$	BB$_{16}$	FD$_{16}$	F5$_{16}$
E6$_{16}$	1B$_{16}$	EE$_{16}$	00$_{16}$	F6$_{16}$	CD$_{16}$	FE$_{16}$	8E$_{16}$
E7$_{16}$	03$_{16}$	EF$_{16}$	00$_{16}$	F7$_{16}$	95$_{16}$	FF$_{16}$	3E$_{16}$

7 Adding More Arithmetic

You can now add several more cases to step **3** of the main procedure. These cases add more arithmetic capabilities to the machine.

3 If k is

24$_{16}$ then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP y.
- **3** POP x.
- **4** If $x < y$,
  then PUSH $2^{64} - 1$,
  otherwise PUSH 0.

2C$_{16}$ then

- **1** Initialize the temporary 64-bit element x to 0.
- **2** POP x.
- **3** If $x < 64$,
  then PUSH 2^x,
  otherwise PUSH 0.

Test 13. To test the machine, set the following elements of M to the provided values and start the machine.

$\mathtt{00}_{16}$	$\mathtt{08}_{16}$	$\mathtt{08}_{16}$	$\mathtt{24}_{16}$	$\mathtt{10}_{16}$	$\mathtt{13}_{16}$	$\mathtt{18}_{16}$	$\mathtt{2C}_{16}$	$\mathtt{20}_{16}$	$\mathtt{2C}_{16}$	$\mathtt{28}_{16}$	$\mathtt{00}_{16}$
$\mathtt{01}_{16}$	$\mathtt{24}_{16}$	$\mathtt{09}_{16}$	$\mathtt{13}_{16}$	$\mathtt{11}_{16}$	$\mathtt{08}_{16}$	$\mathtt{19}_{16}$	$\mathtt{08}_{16}$	$\mathtt{21}_{16}$	$\mathtt{00}_{16}$	$\mathtt{29}_{16}$	$\mathtt{00}_{16}$
$\mathtt{02}_{16}$	$\mathtt{13}_{16}$	$\mathtt{0A}_{16}$	$\mathtt{08}_{16}$	$\mathtt{12}_{16}$	$\mathtt{24}_{16}$	$\mathtt{1A}_{16}$	$\mathtt{23}_{16}$	$\mathtt{22}_{16}$	$\mathtt{40}_{16}$	$\mathtt{2A}_{16}$	$\mathtt{00}_{16}$
$\mathtt{03}_{16}$	$\mathtt{08}_{16}$	$\mathtt{0B}_{16}$	$\mathtt{25}_{16}$	$\mathtt{13}_{16}$	$\mathtt{13}_{16}$	$\mathtt{1B}_{16}$	$\mathtt{10}_{16}$	$\mathtt{23}_{16}$	$\mathtt{22}_{16}$	$\mathtt{2B}_{16}$	$\mathtt{00}_{16}$
$\mathtt{04}_{16}$	$\mathtt{24}_{16}$	$\mathtt{0C}_{16}$	$\mathtt{13}_{16}$	$\mathtt{14}_{16}$	$\mathtt{24}_{16}$	$\mathtt{1C}_{16}$	$\mathtt{2C}_{16}$	$\mathtt{24}_{16}$	$\mathtt{00}_{16}$	$\mathtt{2C}_{16}$	$\mathtt{A0}_{16}$
$\mathtt{05}_{16}$	$\mathtt{13}_{16}$	$\mathtt{0D}_{16}$	$\mathtt{24}_{16}$	$\mathtt{15}_{16}$	$\mathtt{08}_{16}$	$\mathtt{1D}_{16}$	$\mathtt{08}_{16}$	$\mathtt{25}_{16}$	$\mathtt{00}_{16}$		
$\mathtt{06}_{16}$	$\mathtt{24}_{16}$	$\mathtt{0E}_{16}$	$\mathtt{08}_{16}$	$\mathtt{16}_{16}$	$\mathtt{22}_{16}$	$\mathtt{1E}_{16}$	$\mathtt{24}_{16}$	$\mathtt{26}_{16}$	$\mathtt{00}_{16}$		
$\mathtt{07}_{16}$	$\mathtt{08}_{16}$	$\mathtt{0F}_{16}$	$\mathtt{25}_{16}$	$\mathtt{17}_{16}$	$\mathtt{10}_{16}$	$\mathtt{1F}_{16}$	$\mathtt{10}_{16}$	$\mathtt{27}_{16}$	$\mathtt{00}_{16}$		

When the machine terminates, the value of S should be $\mathtt{D0}_{16}$, and the following elements of M should have the indicated values.

$\mathtt{D0}_{16}$	$\mathtt{01}_{16}$	$\mathtt{D8}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E0}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E8}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F0}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{F8}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D1}_{16}$	$\mathtt{00}_{16}$	$\mathtt{D9}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E1}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E9}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F1}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{F9}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D2}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DA}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E2}_{16}$	$\mathtt{00}_{16}$	$\mathtt{EA}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F2}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FA}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D3}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DB}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E3}_{16}$	$\mathtt{00}_{16}$	$\mathtt{EB}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F3}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FB}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D4}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DC}_{16}$	$\mathtt{04}_{16}$	$\mathtt{E4}_{16}$	$\mathtt{00}_{16}$	$\mathtt{EC}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F4}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FC}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D5}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DD}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E5}_{16}$	$\mathtt{00}_{16}$	$\mathtt{ED}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F5}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FD}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D6}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DE}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E6}_{16}$	$\mathtt{00}_{16}$	$\mathtt{EE}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F6}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FE}_{16}$	$\mathtt{00}_{16}$
$\mathtt{D7}_{16}$	$\mathtt{00}_{16}$	$\mathtt{DF}_{16}$	$\mathtt{00}_{16}$	$\mathtt{E7}_{16}$	$\mathtt{00}_{16}$	$\mathtt{EF}_{16}$	$\mathtt{00}_{16}$	$\mathtt{F7}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{FF}_{16}$	$\mathtt{00}_{16}$

8 Adding Bitwise Boolean Logic

You can now add several more cases to step **3** of the main procedure. These cases add bitwise Boolean logic capabilities to the machine.

3 If k is

$\mathtt{28}_{16}$ then

- **1** Initialize the temporary 64-bit elements x, y, and z to 0.
- **2** POP y.
- **3** POP x.
- **4** For every $i \in \{0,\ldots,63\}$, if $x.\text{bit}[i]$ is 1 and $y.\text{bit}[i]$ is 1,
  then set $z.\text{bit}[i]$ to 1,
  otherwise set $z.\text{bit}[i]$ 0.
- **5** PUSH z.

$\mathtt{29}_{16}$ then

- **1** Initialize the temporary 64-bit elements x, y, and z to 0.
- **2** POP y.
- **3** POP x.
- **4** For every $i \in \{0,\ldots,63\}$, if $x.\text{bit}[i]$ is 0 and $y.\text{bit}[i]$ is 0,
  then set $z.\text{bit}[i]$ to 0,
  otherwise set $z.\text{bit}[i]$ to 1.
- **5** PUSH z.

$\mathtt{2A}_{16}$ then

- **1** Initialize the temporary 64-bit elements x and z to 0.
- **2** POP x.
- **3** For every $i \in \{0,\ldots,63\}$, if $x.\text{bit}[i]$ is 1,
  then set $z.\text{bit}[i]$ to 0,
  otherwise set $z.\text{bit}[i]$ to 1.

4 PUSH z.

$2B_{16}$ then

1 Initialize the temporary 64-bit elements x, y, and z to 0.
2 POP y.
3 POP x.
4 For every $i \in \{0,\ldots,63\}$, if x.bit$[i]$ is different from y.bit$[i]$,
 then set z.bit$[i]$ to 1,
 otherwise set z.bit$[i]$ to 0.
5 PUSH the 64-bit value z.

Test 14. To test the machine, set the following elements of M to the provided values and start the machine.

00_{16}	08_{16}	08_{16}	$1A_{16}$	10_{16}	13_{16}	18_{16}	$2A_{16}$	20_{16}	77_{16}	28_{16}	17_{16}
01_{16}	$1A_{16}$	09_{16}	13_{16}	11_{16}	08_{16}	19_{16}	00_{16}	21_{16}	FF_{16}	29_{16}	$3F_{16}$
02_{16}	13_{16}	$0A_{16}$	08_{16}	12_{16}	22_{16}	$1A_{16}$	98_{16}	22_{16}	88_{16}		
03_{16}	08_{16}	$0B_{16}$	22_{16}	13_{16}	13_{16}	$1B_{16}$	$E7_{16}$	23_{16}	60_{16}		
04_{16}	22_{16}	$0C_{16}$	13_{16}	14_{16}	$2B_{16}$	$1C_{16}$	$D9_{16}$	24_{16}	09_{16}		
05_{16}	13_{16}	$0D_{16}$	29_{16}	15_{16}	08_{16}	$1D_{16}$	58_{16}	25_{16}	$5C_{16}$		
06_{16}	28_{16}	$0E_{16}$	08_{16}	16_{16}	$1A_{16}$	$1E_{16}$	$1B_{16}$	26_{16}	$7D_{16}$		
07_{16}	08_{16}	$0F_{16}$	$1A_{16}$	17_{16}	13_{16}	$1F_{16}$	$C9_{16}$	27_{16}	$2C_{16}$		

When the machine terminates, the value of S should be $E0_{16}$, and the following elements of M should have the indicated values.

$E0_{16}$	67_{16}	$E8_{16}$	10_{16}	$F0_{16}$	98_{16}	$F8_{16}$	88_{16}
$E1_{16}$	18_{16}	$E9_{16}$	87_{16}	$F1_{16}$	$E7_{16}$	$F9_{16}$	60_{16}
$E2_{16}$	26_{16}	EA_{16}	$D0_{16}$	$F2_{16}$	$D9_{16}$	FA_{16}	09_{16}
$E3_{16}$	$A7_{16}$	EB_{16}	04_{16}	$F3_{16}$	$5C_{16}$	FB_{16}	58_{16}
$E4_{16}$	$E4_{16}$	EC_{16}	66_{16}	$F4_{16}$	$7F_{16}$	FC_{16}	19_{16}
$E5_{16}$	36_{16}	ED_{16}	$E5_{16}$	$F5_{16}$	ED_{16}	FD_{16}	08_{16}
$E6_{16}$	88_{16}	EE_{16}	60_{16}	$F6_{16}$	77_{16}	FE_{16}	17_{16}
$E7_{16}$	00_{16}	EF_{16}	$C0_{16}$	$F7_{16}$	FF_{16}	FF_{16}	$3F_{16}$

9 Joining the Machine With the Devices

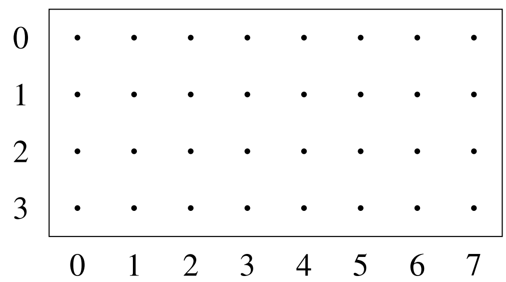
At this point you should have a machine with fully functional computational capabilities. What you need to do now is to join the machine with the devices that allow it to consume data from its environment and to produce data to its environment. There are four devices:

- The *Image Input* device allows the machine to consume images as a two-dimensional array of gray-scale values. This device is very important, because it is what enables the machine to load programs encoded as images on the film.
- The *Image Output* device allows the machine to produce color images. Moving images are supported by producing a time series of still images.
- The *Audio Output* device allows the machine to produce audio signals as a time series of amplitude values.
- The *Text Output* device allows the machine to produce text as a stream of characters.

The descriptions below will only explain the correspondence between machine events and external events. It is up to you to make sure that the interpretation of external events is faithfully implemented. As before, you will add more cases to the existing machine. Wherever external interaction is required, this is written ⟨like this⟩.

9.1 Image Input

The *Image Input* device allows the machine to consume an image as a two-dimensional array of points of light intensity values. The following figure shows an example of such an array, consisting of 32 sampling points arranged in 8 columns and 4 rows.



As shown, both columns and rows are numbered consecutively, starting at 0. The spacing between the sampling points must be uniform in both horizontal and vertical directions, and an anti-aliasing filter must be employed to limit the bandwidth of the image to satisfy the Nyquist-Shannon sampling theorem.

Each picture element detects the intensity of light transmitted or reflected at a sampling point in that particular position of the image, represented as one of 256 intensity levels, from 0 (minimum intensity) to 255 (maximum intensity). Values between 0 and 255 represent intermediate intensities between these extremes.

In order to implement the image input device, you will need to add two extra cases to step **3** of the main procedure.

3 If k is

FF_{16} then

- **1** Initialize the temporary 64-bit elements c and r to 0.
- **2** ⟨Ready the next image to be consumed by the machine.⟩
- **3** ⟨Find the number of columns and rows in the image.⟩
- **4** Set c to the number of columns in the image.
- **5** PUSH c.
- **6** Set r to the number of rows in the image.
- **7** PUSH r.

FE_{16} then

- **1** Initialize the temporary 64-bit elements x and y to 0.
- **2** POP x.
- **3** POP y.
- **4** ⟨Measure the intensity of light at column x and row y in the image.⟩
- **5** Set z to the intensity level of light in the image at column x and row y.
- **6** PUSH z.

9.2 Image Output

The *Image Output* device allows the machine to produce an image represented as a two-dimensional array of points of color space values. Moving images can be produced as a sequence of images.

In order to implement the image output device, you will need to add two extra cases to step **3** of the main procedure.

3 If k is

`FD`$_{16}$ then

1 Finish and render the frame constructed so far.
2 Initialize the temporary 64-bit elements r, w, and h to 0.
3 POP r.
4 POP h.
5 POP w.
6 ⟨Set the audio sample rate to r, the width of the frame to w, and the height of the frame to h.⟩

`FC`$_{16}$ then

1 Initialize the temporary 64-bit elements x, y, r, g, and b, to 0.
2 POP b.
3 POP g.
4 POP r.
5 POP y.
6 POP x.
7 ⟨Set the picture element at column x and row y to the point in the color space represented by the tuple (r, g, b).⟩

9.3 Audio Output

The *Audio Output* device allows the machine to produce a two-channel audio signal encoded digitally using Linear Pulse Code Modulation. The device must create an audio signal passing through a series of magnitude values specified by the program. The bandwith of this audio signal must be less than half of the sampling frequency. Each channel value is in the range $\{0,\dots,2^{16}-1\}$.

In order to implement the audio output device, you will need to add one extra case to step **3** of the main procedure.

3 If k is

`FB`$_{16}$ then

1 Initialize the temporary 64-bit elements l and r to 0.
2 POP r.
3 POP l.
4 ⟨Set the audio signal magnitude of the left channel to l.⟩
5 ⟨Set the audio signal magnitude of the right channel to r.⟩

9.4 Text Output

In order to implement the text output device, you will need to add one extra case to step **3** of the main procedure.

3 If k is

`FA`$_{16}$ then

1 Initialize the temporary 64-bit element c to 0.
2 POP c.

3 $\langle$Produce the character with Unicode code point c.$\rangle$

9.5 Octet Output

In order to implement the octet output device, you will need to add one extra case to step **3** of the main procedure.

3 If k is

$\texttt{F9}_{16}$ then

1 Initialize the temporary 64-bit element x to 0.

2 Initialize the temporary 8-bit element o to 0.

3 POP x.

4 For all $i \in \{0, \ldots, 7\}$ set $o.\mathrm{bit}[i]$ to $x.\mathrm{bit}[i]$.

5 $\langle$Produce the number o.$\rangle$

10 Installing the Initial Program

You first need to obtain the intial program from elsewhere on the film as a sequence of p octets in hexadecimal notation. These octets must be stored in $M[i]$, where $i \in \{0, \ldots, p-1\}$.

The operation of the initial program, and programs loaded by the initial program, can be controlled by a set of parameters represented as a sequence of q octets. These octets must also be stored in M. This is done as follows:

1 PUT 8 octets from q into index p.

2 Store the sequence of parameters in elements $M[p+8]$ to $M[p+8+q-1]$.

When these things have been done, the machine can be started, and it can then load a program from the film and execute it.

A Number Notation

A.1 Binary Notation

Each element contains a value represented as a sequence of binary digits. In this documentation, individual binary digits are referred to using non-negative integers, listed in decreasing order. For an n-bit value, the individual bits, b_i, are numbered as follows:

$$b_{n-1}\; b_{n-2}\; \ldots\; b_1\; b_0$$

When an n-bit binary element is interpreted as a non-negative integer, the value of that integer is

$$v = \sum_{i=0}^{n-1} b_i 2^i$$

A.2 Hexadecimal Notation

For convenience, binary numbers are normally written in hexadecimal notation. Each hexadecimal digit corresponds to a group of four binary digits, as shown in the following table.

Hexadecimal digit	Binary digits
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Hexadecimal digit	Binary digits
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111