PROJECT ON THE COMPUTER IMPLEMENTATION OF $GF(2^n)$ ARITHMETIC

SAMUEL J. LOMONACO, JR.

The objective of this project is to learn how to implement the arithmetic operations of the characteristic 2 finite field $GF(2^n)$ on a computer with fixed wordlength L, where $n < L-1$. On such a machine, a computer word w consists of L bits $w_{L-1}, \ldots, w_2, w_1, w_0$, listed from right to left with the indices $0, 1, 2, \ldots, L-1$, and diagrammatically represented as

$$\begin{array}{rc|c|c|c|c|c|} \text{Bit Index} & & L-1 & \cdots & 2 & 1 & 0 \\ \cline{3-7} \text{Word} & w = & w_{L-1} & \cdots & w_2 & w_1 & w_0 \\ \cline{3-7} \end{array}$$

We assume that the computer is equipped with the following inline functions:

- **Bitwise exclusive 'OR'**, denoted by XOR, and defined as:
$$\mathrm{XOR}(w,u) = (w_{L-1}+u_{L-1}, \ldots, w_2+u_2, w_1+u_1, w_0+u_0) \ ,$$
where w and u are computer words, and where '+' denotes exclusive 'OR', a.k.a., as addition mod 2.
- **Left Shift**, denoted by LSHIFT, and defined as:
$$\mathrm{LSHIFT}(w) = (w_{L-2}, \ldots, w_2, w_1, w_0, 0) \ ,$$
where w denotes a computer word.
- **Bit**, denoted by BIT, and defined by
$$\mathrm{BIT}(j,w) = w_j \ ,$$
where again w denotes a computer word.
- **Set Bit**, denoted by SETBIT(j,w), when called sets the j-th bit w_j of the the computer word w to 1, i.e., sets $w_j = 1$.

Let
$$\widetilde{p} = x^n + p_{n-1}x^{n-1} + \cdots + p_2x^2 + p_1x + p_0$$
be a degree n primitive binary polynomial defining the Galois field $GF(2^n)$, i.e.,
$$GF(2^n) = GF(2)[x]/(p) \ ,$$
and let ξ denote the primitive root
$$\xi = x + (p) \ .$$

We represent the primitive polynomial $\widetilde{p}$ as the computer word

$$\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c} & L-1 & \cdots & n+2 & n+1 & n & n-1 & \cdots & 2 & 1 & 0 & \\ \cline{2-11} \widetilde{p} = & 0 & \cdots & 0 & 0 & 1 & p_{n-1} & \cdots & p_2 & p_1 & p_0 & , \\ \cline{2-11} \end{array}$$

Date: March 5, 2012.

and we represent each element $a = \sum_{j=0}^{n-1} a_j \xi^j$ of the field $GF\left(2^{n-1}\right)$ as the computer word

$$a = \begin{array}{c} L-1 \quad \cdots \quad n+1 \quad n \quad n-1 \quad \cdots \quad 2 \quad 1 \quad 0 \\ \begin{array}{|c|c|c|c|c|c|c|c|c|} \hline 0 & \cdots & 0 & 0 & a_{n-1} & \cdots & a_2 & a_1 & a_0 \\ \hline \end{array} \end{array} .$$

Caveate: *The reader should take care to note that we are using two different interpretations of computer words in this assignment, i.e., one as an element of the Galois Field $GF\left(2^n\right)$, the other as an element the polynomial ring $GF(2)[x]$. Which interpretation is chosen is determined by context. As an aid to the reader, we use a to denote the element of the finite field $GF\left(2^n\right)$, and $\widetilde{a}$ to denote the element of the polynomial ring $GF(2)[x]$. Thus,*

$$a = \sum_{j=0}^{n-1} a_j \xi^j \quad \text{and} \quad \widetilde{a} = \sum_{j=0}^{n-1} a_j x^j \ .$$

To avoid confusion, it is suggested that, at each step, the reader ask the question: **"Where does the object 'live' " ?**

The procedure ADD(a, b) for addition '$+$' in $GF\left(2^n\right)$ is simply the inline routine XOR(a, b)

We now construct an algorithmic procedure MULT(a, b) for multiplication '$\cdot$' in $GF\left(2^n\right)$. We begin by noting that

$$ab = \left(\sum_{j=0}^{n-1} a_j \xi^j\right) b = \sum_{j=0}^{n-1} a_j \left(\xi^j b\right) \quad .$$

So to implement MULT, we first need to construct a subroutine XISHIFT(b) that computes and returns ξb .

```
PROC XISHIFT(b)
LOCAL B
   B = LSHIFT(b)
   IF BIT(n, B) = 1 THEN
      B = XOR(B, p)
   END IF
   RETURN(B)
END PROC
```

We next create MULT by iterately using XISHIFT to compute $\xi^j b$ and then XORing each such term in whenever $a_j \neq 0$. Thus, we have:

```
PROC MULT(a, b)
LOCAL A, B, C, j
   A = a
   B = b
   C = 0
   LOOP j = 0..n DO
      IF BIT(j, A) = 1 THEN
         C = XOR(C, B)
```

End If
$B = $ XiShift(B)
End Loop
Return(C)
End Proc

Our next objective is to construct an algorithmic procedure Inverse(a) that finds the inverse a^{-1} of each element a in $GF\left(2^n\right)$, provided $a \neq 0$. We will do so as follows:

First note that, since the primitive polynomial $\widetilde{p}$ is irreducible, it follows that $\widetilde{p}$ is relatively prime to every non-zero polynomial $\widetilde{a}$ of degree less than n, i.e.,

$$\gcd\left(\widetilde{a}, \widetilde{p}\right) = 1 \ .$$

Next note that the extended Euclidean algorithm finds polynomials $\widetilde{s}$ and $\widetilde{t}$ such that

$$\widetilde{a}\widetilde{s} + \widetilde{p}\widetilde{t} = 1 \ .$$

Thus, s is the inverse a^{-1} of a in $GF\left(2^n\right)$.

Your project consists of the following three problems:

Problem 1. Let $\widetilde{a}$ and $\widetilde{b}$ be polynomials in $GF(2)[x]$, and let $\widetilde{q}$ and $\widetilde{r}$ be the corresponding unique polynomials in $GF\left(2^n\right)$ such that

$$\widetilde{a} = \widetilde{b}\widetilde{q} + \widetilde{r} \quad ,$$

where $\widetilde{r} = 0$ or $\deg\left(\widetilde{r}\right) < \deg\left(\widetilde{b}\right)$. Using the programming language C, construct two subroutines Quo$\left(\widetilde{a}, \widetilde{b}\right)$ and Rem$\left(\widetilde{a}, \widetilde{b}\right)$ which respectively compute $\widetilde{q}$ and $\widetilde{r}$. If these subroutines are available as inline subroutines, then simply use the inline routines.

Problem 2. Using the above algorithmic procedures Quo$\left(\widetilde{a}, \widetilde{b}\right)$ and Rem$\left(\widetilde{a}, \widetilde{b}\right)$, construct in C an algorithmic procedure Inverse(s) which computes the inverse a^{-1} of a in $GF\left(2^n\right)$, provided $a \neq 0$.

Problem 3. Implement in C, a subroutine Power(a, ℓ) which takes as input an element $a \in GF\left(2^n\right)$ and a positive integer ℓ, and then produces as output $a^\ell \in GF\left(2^n\right)$. Be sure to compute a^ℓ by the method of repeated squaring.

Remark 1. *Be sure to create a C subroutine* SetEnvironment$(n, \widetilde{p})$ *which sets the positive integer n and the primitive polynomial $\widetilde{p}$ as global variables.*

University of Marland Baltimore County
E-mail address: Lomonaco@umbc.edu
URL: www.csee.umbc.edu/~lomonaco