UNIVERZA V LJUBLJANI
Fakulteta za računalništvo in informatiko

# LFSR

## (Linear Feedback Shift Register)

**Tečaj iz kriptografije in računalniške varnosti**
**Predavatelj: Aleksandar Jurišič**

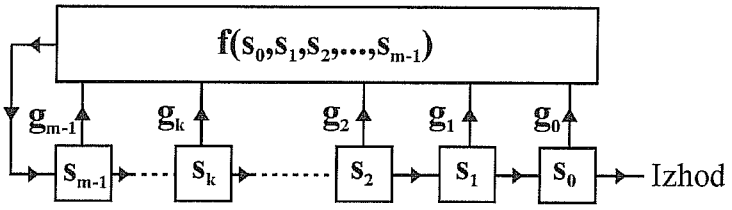
**Robert Kuster**

**Ljubljana, 2002**

# 0. Kazalo

# 1. Uvod

Algoritma, ki bi na stroju s končnim številom stanj lahko generiral povsem naključna števila ni. Ravno končnost stroja nas namreč prisili, da so generirane bitne sekvence periodične. Najboljše kar lahko dosežemo so sekvence z zelo dolgo periodo, imenovane **psevdonaključne** sekvence. LFSR je jedro vsakega digitalnega sistema, ki temelji na psevdonaključnih številih ali bitnih zaporedjih. Sestavljen je iz preprostega pomikalnega registra in nekaj (odvisno od števila povratnih povezav) XOR vrat:




*Slika 1: Shema LFSR generatorja*

Pri tem je $f$ linearna funkcija $f(s) = \sum_{i=0}^{m-1} g_i s_i$ .

Izhod iz LFSR generatorja je določen z začetnim stanjem registra $s_0, s_1, ..., s_{m-1}$ in z rekurzivno relacijo $s_{k+m} = \sum_{i=0}^{m-1} g_i s_{k+i}, \quad k \geq 0$ .

V nadaljevanju si bomo pogledali dve vrsti implementacije LFSR in izbiro povratnih povezav, da bo perioda generirane sekvence imela največjo možno dolžino.

# 2. Golombovi principi

Da bi bila neka bitna sekvenca čimbolj podobna naključni sekvenci, morajo biti izpolnjeni sledeči pogoji:

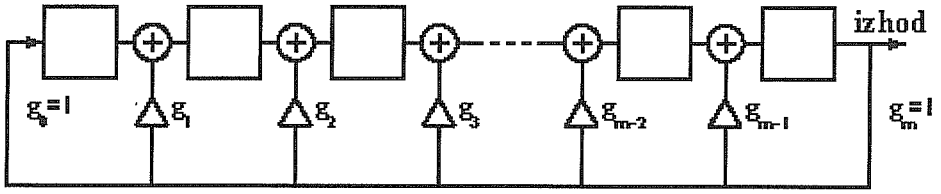
1. Število ničel in enic v periodi naj bo čim bolj enako.
2. Polovica gruč v periodi ima dolžino 1, četrtina gruč ima dolžino 2,..., $1/2^i$ gruč ima dolžino $i$. Polovica gruč katerokoli dolžine je oblike ...01110... (blok), polovica pa oblike...10001... (reža).
3. Izvenfazna avtokorelacija (out-of-phase autocorrelation) $AC(k)$ ima vedno enako vrednost, ne glede na zamik $k$.

   $AC(k) = \frac{št.\ ujemanj \ - \ št.\ razlik}{p}$, pri čemer primerjamo sekvenco (s periodo $p$) in njen zamik za $k$ mest. Avtokorelacija je izvenfazna, če zamik $k$ n deljiv s periodo $p$.

## 3. Izvedbe LFSR generatorjev

LFSR lahko implementiramo na dva načina:

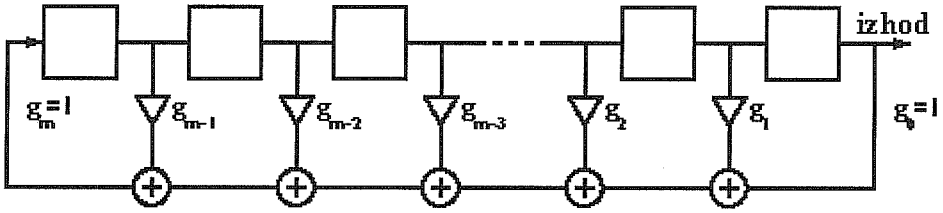
1. **Fibonaccijeva** izvedba: LFSR sestoji iz preprostega pomikalnega registra (shift register), pri čemer so nekatere povratne povezave preko XOR vrat speljane nazaj na vhod.




*Slika 2: Fibonaccijeva izvedba LFSR generatorja*

Če povratne povezave ni, je člen je $g_i = 0$, če obstaja pa je enak 1. $g_0$ in $g_m$ (vhod in izhod iz pomikalnega registra) sta vedno 1.

2. **Galoiseva** izvedba: LFSR sestoji iz pomikalnega registra, pri katerem nekatere vmesne vrednosti XORamo z izhodno vrednostjo.




*Slika 3: Galoiseva izvedba LFSR generatorja*

Iz slik je razvidno, da je ureditev členov $g_i$ pri Galoisovi obratna od tiste pri Fibonaccijevi izvedbi. V praksi se pogosteje uporablja Galoiseva izvedba, ker je zaradi manjšega števila XOR vrat v povratni zanki hitrejša.

Začetna vrednost pri Fibonaccijevi izvedbi se imenuje **inicializacijski vektor** in predstavlja prvih *m* izhodnih bitov oz. prvi *m*-bitov generirane sekvence.

## 4. Označevanje povratnih povezav

Set povratnih povezav podamo tako, da njihove indekse enostavno podamo v oklepaju. Ker je povezava $g_0$ vedno prisotna, se podaja implicitno. Čeprav velja enako tudi za povezavo $g_m$, se ta vseeno navede - iz nje lahko razberemo velikost pomikalnega registra.

Set povratnih povezav za Galoisov generator je podan kot $\left[f_1, f_2, f_3, ..., f_j\right]_g$. Indeks $j$ predstavlja skupno število povratnih povezav (brez $g_0$), $f_1 = m$ pa je velikost pomikalnega registra. Indeks $g$ pomeni, da gre za Galoisov LFSR generator.

Set povratnih povezav za ekvivalenten Fibonaccijev generator je $\left[f_1, m-f_2, m-f_3, ..., m-f_j\right]_f$.

Kot primer vzemimo LFSR velikosti $m = 8$ s povratnimi povezavami $g_8, g_6, g_5, g_4$ in $g_0$. Povratne povezave za Galoisev generator bi tako podali kot $\left[8,6,5,4\right]_g$, za Fibonaccijev generator pa kot $\left[8,8-6,8-5,8-4\right]_f = \left[8,2,3,4\right]_f = \left[8,4,3,2\right]_f$.

V splošnem pa velja pravilo, da je prva povratna povezava izhod iz LFSR generatorja, druga je levo od nje, itd..., ne glede na to, ali gre za Fibonaccijev ali Galoisev LFSR generator naključnih števil. Tak način je uporabljen tudi v tabelah, kjer so podani seti povratnih povezav za PN-sekvence.

## 5. PN-sekvence ali sekvence z maksimalno dolžino

Ker so vse operacije linearne, dobimo z LFSR generatorji takoimenovane linearno rekursivne sekvence (LRS). Perioda generirane sekvence je v splošnem odvisna od dveh stvari: izbire povratnih povezav in začetne vrednosti registra. **Sekvenca generirana z LFSR generatorjem velikosti *m* in periodo $N = 2^m - 1$ se imenuje PN-sekvenca (psevdo-noice) ali sekvenca z maksimalno dolžino.**

Vsak LFSR generator poljubne dolžine *m* lahko ob ustrezni izbiri povratnih povezav generira **PN-sekvenco**. Generator s tako izbranimi povratnimi povezavami lahko dejansko generira dve sekvenci:

- trivialna sekvenca dolžine 1; dobimo jo v primeru, ko register inicializiramo s samimi ničlami.
- »uporabna« sekvenca dolžine $N = 2^m - 1$; dobimo jo v primeru, ko register inicializiramo s katerokoli od nič različno vrednostjo.

Tako dobimo za *m*-bitni register skupaj vseh $2^m$ možnih stanj.

Če izbira povratnih povezav ni taka, da bi LFSR generiral PN-sekvenco, pa je dolžina sekvence odvisna od začetnega stanja registra. Tak LFSR lahko generira dve ali več povsem različnih sekvenc (ne vštevši trivialne s samimi ničlami), odvisno od izbranega začetnega stanja. Z vsemi temi sekvencami skupaj dobimo vseh $2^m$ možnih stanj registra.
V resnici pa se taki »ne-maksimalni« LFSR generatorji uporabljajo redko.

## 6. Nekaj definicij iz algebre

Vsak LFSR lahko predstavimo kot polinom spremenljivke X, imenovan generatorski ali karakteristični polinom: $G(x) = g_m x^m + g_{m-1}x^{m-1} + g_{m-2}x^{m-2} + ... + g_2 x^2 + g_1 x^1 + g_0$.

Koeficienti $g_i$ predstavljajo povratne povezave. Če povratne povezave ni, je koeficient $g_i = 0$, če obstaja, pa je enak 1. Stopnja polinoma $m$ nam predstavlja velikost pomikalnega registra. Za karakteristični polinom veljajo pravila linearne algebre, s tem da so vse operacije izvršene po modulu 2:

| Seštevanje: | Množenje: |
|---|---|
| 0+0=0 | 0*0=0 |
| 0+1=1 | 0*1=0 |
| 1+1=0 | 1*1=1 |

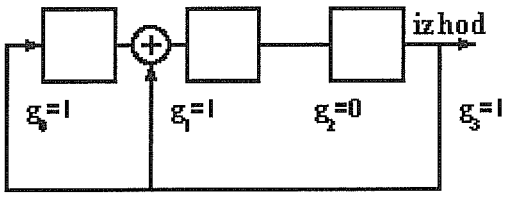
Nekaj dejstev:

1. Vsak polinom $G(x)$ za katerega je $G(0) = 1$, deli polinom $x^p + 1$ za nek $p$. Najmanjši $p$, ki ustreza temu pogoju, se imenuje **perioda** polinoma $G(x)$.
2. **Nerazcepni polinom** (ne moremo ga zapisati kot produkt dveh drugih nekonstantnih polinomov) stopnje $m$ ima periodo, ki deli $N = 2^m - 1$.
3. Nerazcepni polinom stopnje $m$ s periodo $N = 2^m - 1$ se imenuje **primitivni polinom.**

**LFSR generira PN-sekvenco samo tedaj, če je njegov karakteristični polinom primitiven.**

Primeri:

1. Generatorski polinom $G(x) = x^3 + x^1 + 1$ predstavlja LFSR s povratnima povezavama 3 in 1. Konstanta 1 predstavlja vhod v LFSR - $g_0$.




*Slika 4: LFSR (Galois) s karakterističnim polinomom* $G(x) = x^3 + x^1 + 1$

$m = 3 \Rightarrow N = 2^3 - 1 = 7$

$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1)$

Ker je polinom $G(x) = x^3 + x^1 + 1$ nerazcepen in ker je njegova perioda $N = 7$, je $G(x)$ tudi primitiven. To pomeni, da bo LFSR generiral PN-sekvence.

Ob začetni vrednosti registra 100 dobimo:

*Tabela 1: Generirana sekvenca*

| Urin signal | Vsebina registra | Izhod |
|---|---|---|
| 0 | 100 | - |
| 1 | 010 | 0 |
| 2 | 001 | 0 |
| 3 | 110 | 1 |
| 4 | 011 | 0 |
| 5 | 111 | 1 |
| 6 | 101 | 1 |
| 7 | 100 | 1 |

LSFR s karakterističnim polinomom $G(x) = x^3 + x^1 + 1$ in začetno vrednostjo 100 nam torej generira sekvenco 0010111.

2. $G(x) = x^4 + x^3 + x^2 + 1 = (x+1)(x^3 + x + 1)$.
Polinom $G(x)$ lahko razcepimo, kar pomeni da ni primitiven.

3. $G(x) = x^4 + x^3 + x^2 + x + 1$.
Polinom je sicer nerazcepen, ker pa je njegova perioda 5 ($x^5 + 1 = (x+1) \cdot G(x)$), ni primitiven.

4. $G(x) = x^4 + x^3 + 1$
Polinom $G(x)$ je nerazcepen. Da bi določili njegovo perodo, moramo določiti najmanjši $p$, tako da bo $x^p + 1$ deljiv z $G(x)$. Gotovo bo $p > 4$ in zaradi 2) bo $p$ delitelj števila $N = 2^4 - 1 = 15$ → perioda $p$ je lahko 5 ali 15.
$x^5 + 1 = (x+1)(x^4 + x^3 + 1) + (x^3 + x)$
$x^{15} + 1 = (x^{11} + x^{10} + x^9 + x^8 + x^6 + x^4 + x^3 + 1)(x^4 + x^3 + 1)$
→ $G(x)$ ima periodo 15 in je zato primitiven.

V prejšnjih primerih smo preverjali, če bo nek set povratnih povezav generiral PN-sekvenco. Največkrat pa delamo ravno obratno – iščemo vse sete povratnih povezav, ki bi generirali PN-sekvenco za določeno velikost registra.
Če bi npr. iskali vse take sete za register velikosti *m=3*, storimo to tako: dolžina sekvence bo $N = 2^3 - 1 = 7$. Vemo, da rešitev leži v vseh primitivnih faktorjih polinoma $x^7 + 1$. Z uporabo linearne algebre po modulu 2 dobimo:
$x^7 + 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1)$
Primitivni polinomi so tisti, katerih stopnja je enaka velikosti registra, torej 3. Tako obstajata dva seta povratnih povezav [3,1] in [3,2], pri katerih bo LFSR z velikostjo registra *m=3* generiral PN-sekvence.

V bistvu je številu setov, s katerimi bo nek LFSR poljubne velikosti $m$ generiral PN-sekvence vedno sodo. Še več: seti se vedno pojavljajo v »zrcalnih« parih - če imamo set $[f_1, f_2, f_3, ..., f_j]$, bo vedno obstajal tudi set $[f_1, m - f_2, m - f_3, ..., m - f_j]$, ki je zrcalna slika originalnega seta. Tudi sekvenca generirana s takim setom bo zrcalna slika originalne sekvence.

## 7. Psevdonaključnost PN-sekvenc

G1: Ker se vsako neničelno stanje registra pojavi natanko enkrat in ker je najbolj desni bit tega stanja naslednji izhodni bit, vidimo, da je v vsaki periodi natanko $2^{m-1}$ enic in $2^{m-1}-1$ ničel.

G2: V sekvenci se bo pojavila gruča (bodisi blok bodisi vrzel) dolžine $k$ ($k \le m-2$) samo v primeru, če je desnih $k+2$ bitov registra oblike 011...1110 ali 100...0001. Ker gre register skozi vsa neničelna stanja, je skupno število gruč (ene ali druge oblike) z dolžino $k$ enako $2^{m-k-2}$. Obstaja eno stanje 11..110, ki mu sledi 11..11, temu pa 011..11. Tako obstaja en blok dolžine $m$ in noben dolžine $m-1$. Podobno ni nobene reže velikosti $m$ ter ena velikost $m-1$.
Skupno število vseh gruč je potemtakem:

$$2\sum_{k=1}^{m-2} 2^{m-k-2} + 2 = 2\left(2^{m-2}-1\right)+2 = 2^{m-1}$$

Od tega je $1/2^k$ gruč dolžine $k$.

G3: Naj bo $\{s_i\}$ PN-sekvenca in $\{s_{i+k}\}$ ista sekvenca zamaknjena za $k$ mest. Če obe sekvenci seštejemo, dobimo prav tako PN-sekvenco. Mesta, kjer se $\{s_i\}$ in $\{s_{i+k}\}$ ujemata, bodo imela vrednost 0, ostala pa 1. Upoštevajoč G1 dobimo

$$AC(k) = \frac{\text{št. ujemanj - št. razlik}}{p} = \frac{\left(2^{m-1}-1\right)-\left(2^{m-1}\right)}{2^m-1} = \frac{-1}{2^m-1} \qquad \text{za vse } 1 \le k \le 2^m-1$$

PN-sekvence torej zadostijo vsem Golombovim principom psevdonaključnosti.

## 8. Uporaba v kriptografiji in praksi

### 8.1 Šifriranje in dešifriranje

Psevdonaključne vrednosti generirane s LFSR generatorjem lahko uporabimo za šifriranje in dešifriranje podatkov v komunikacijskih sistemih. Tok digitalnih podatkov enostavno XORamo z izhodom iz LFSR generatorja – tako dobimo šifriran signal. Isto naredimo tudi za dešifriranje, pri čemer moramo paziti le na to, da je začetna vrednost registra v obeh primerih enaka.

Za praktično uporabnost v kriptografiji pa bi morali biti izpolnjeni tudi sledeči pogoji:

- perioda psevdonaključne sekvence mora biti zelo velika (vsaj $\sim 10^{50}$)
  Temu pogoju zadostimo z izbiro dovolj velikega registra. Če vzamemo register velikosti $m = 166$, dobimo periodo $2^{166}-1 > 10^{50}$.
- generiranje sekvence mora biti preprosto (zaradi hitrega šifriranja/dešifriranja).
  Ker je LFSR sestavljen iz preprostih (booleanovih) elementov, je implementacija enostavna delovanje pa izredno hitro.
- kriptosistem mora biti varen proti napadu z izbranim čistopisom
  Če imamo na razpolago $2m$ sledečih si bitov čistopisa $s_k, s_{k+1}, \ldots, s_{k+2m-1}$, lahko zapišemo sistem $m$ enačb z $m$ neznankami $g_0, \ldots, g_{m-1}$, ki ima enolično rešitev. Na ta način lahko dobimo karakteristični polinom $G(x)$.

Kriptosistem baziran na LFSR generatorju naključnih števil torej ni varen proti napadu s poznanim čistopisom. Njegova uporaba v kriptografiji se zato odsvetuje.

### 8.2 Psevdonaključna števila

Mnogi računalniški programi (razne igre, grafični programi, itd...) potrebujejo naključna števila. S pomočjo LFSR generatorjev lahko generiramo psevdonaključna števila. Ker pa mnoge računalniške aplikacij potrebujejo določeno mero ponovljivosti, imajo psevdonaključna števila v določenih situacijah celo prednost pred »pravimi« naključnimi števili. Uporabnik pa mora imeti možnost, da vpliva na začetno vrednost registra – tako lahko dobi več alternativnih psevdonaključnih sekvenc.

### 8.3 LFSR števci

LFSR generatorje lahko uporabimo tudi kot alternativo konvecionalnim binarnim števcem. Prednost je predvsem v enostavnejšem logičnem vezju, slabost v določenih situacijah pa je lahko to, da generirana bitna sekvenca ni urejena po velikosti, kot pri konvencionalnem binarnem števcu. Za implementacijo LFSR števca ponavadi izberemo tak polinom, ki generira PN-sekvenco. Če hočemo, da bo števna perioda različna od $2^m - 1$, vgradimo dodatno logiko, ki »pazi« na vsebino registra in ga ob določeni vrednosti resetira v začetno stanje.

Primer:
Hočemo števec s periodo 12 → rabimo polinom četrte stopnje ($2^4 - 1 = 15$) → ko register vsebuje vrednost 12. stanja, ga resetiramo.

Za karakteristični polinom $G(x) = x^4 + x^3 + 1$ in začetno vrednost 0001 dobimo rezultate prikazane v tabeli 2.

*Tabela 2: Generirana sekvenca*

| stanje | Vsebina registra | stanje | Vsebina registra | |
|---|---|---|---|---|
| 1 | 0001 | 9 | 0101 | |
| 2 | 1000 | 10 | 1010 | |
| 3 | 0100 | 11 | 1101 | |
| 4 | 0010 | **12** | **1110** | ← reset registra |
| 5 | 1001 | 13 | 1111 | |
| 6 | 1100 | 14 | 0111 | |
| 7 | 0110 | 15 | 0011 | |
| 8 | 1011 | | | |

Register resetiramo, ko vsebuje vrednost 1110.

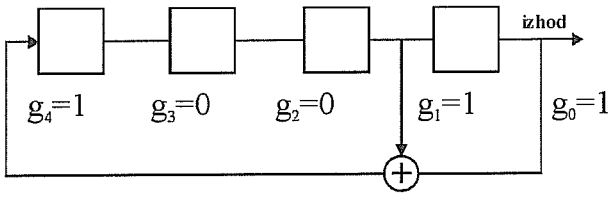
## 9. Reference

[1] http://www.newwaveinstruments.com/resources/articles/m_sequence_linear_feedback_shift_register_lfsr.htm
[2] http://archives.e-insite.net/archives/ednmag/reg/1996/010496/01df4.htm
[3] http://www-math.cudenver.edu/~wcherowi/courses/m5410/m5410fsr.html
[4] http://archives.e-insite.net/archives/ednmag/reg/1998/052198/11df_06.htm

# 10. Priloga

Karakteristični polinom: $G(x) = x^4 + x^3 + 1$
Inicializacijska vrednost registra: 0001

1. Fibonaccijeva izvedba



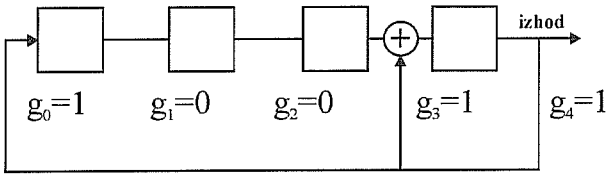

*Slika 5: LFSR (Fibonacci) s karakterističnim polinomom* $G(x) = x^4 + x^3 + 1$

```
/* lfsr-f.c
** Robert Kuster
** Program to print out the sequence and register states
** of the 4-bit LFSR (Fibonacci implementation) used as a counter.
*/

#include <stdio.h>
#include <stdlib.h>


#define BIT(n,x) ( ((x) >> (n)) & 1)    // get bit n of value x

void main()
{
    FILE *file = fopen ("sequence.txt", "w+");
    FILE *file1 = fopen ("register.txt", "w+");

    unsigned reg = 1;
    BOOL feedback = 0;
    int i=0;

    do {
        fprintf (file1,"%.2d %d%d%d%d\n",i++, BIT(3,reg),BIT(2,reg),BIT(1,reg),BIT(0,reg));

        feedback = BIT(0,reg)^BIT(1,reg);
        reg = (reg>>1) + (feedback<<3);
        reg &= 0xF;

        fprintf (file,"%d", feedback);
    } while (reg != 1);

    fclose (file);
    fclose (file1);
}
```

*Tabela 3: Rezultati*

| Urin signal | Vsebina registra | Izhod | Urin signal | Vsebina registra | Izhod |
|---|---|---|---|---|---|
| 0 | 0001 | - | 8 | 0101 | 1 |
| 1 | 1000 | 1 | 9 | 1010 | 1 |
| 2 | 0100 | 0 | 10 | 1101 | 0 |
| 3 | 0010 | 0 | 11 | 1110 | 1 |
| 4 | 1001 | 0 | 12 | 1111 | 0 |
| 5 | 1100 | 1 | 13 | 0111 | 1 |
| 6 | 0110 | 0 | 14 | 0011 | 1 |
| 7 | 1011 | 0 | | | 1 |

## 2. Galoiseva izvedba




*Slika 6: LFSR (Galois) s karakterističnim polinomom* $G(x) = x^4 + x^3 + 1$

```
/* lfsr-g.c
** Robert Kuster
** Program to print out the sequence and register states
** of the 4-bit LFSR (Galois implementation) used as a counter.
*/

#include <stdio.h>
#include <stdlib.h>

#define BIT(n,x) ( ((x) >> (n)) & 1)    // get bit n of value x

void main()
{
    FILE *file = fopen ("sequence.txt", "w+");
    FILE *file1 = fopen ("register.txt", "w+");

    unsigned reg = 1;
    BOOL feedback = 0;
    int i=0;

    do {
        fprintf (file1,"%.2d %d%d%d%d\n",i++, BIT(3,reg),BIT(2,reg),BIT(1,reg),BIT(0,reg));

        feedback = BIT(0,reg);
        reg ^= (feedback<<1);
        reg = (reg>>1) + (feedback<<3);
        reg &= 0xF;

        fprintf (file,"%d", feedback);
    } while (reg != 1);

    fclose (file);
    fclose (file1);
}
```

*Tabela 4: Rezultati*

| Urin signal | Vsebina registra | Izhod | Urin signal | Vsebina registra | Izhod |
|---|---|---|---|---|---|
| 0 | 0001 | - | 8 | 1011 | 1 |
| 1 | 1001 | 1 | 9 | 1100 | 1 |
| 2 | 1101 | 1 | 10 | 0110 | 0 |
| 3 | 1111 | 1 | 11 | 0011 | 0 |
| 4 | 1110 | 1 | 12 | 1000 | 1 |
| 5 | 0111 | 0 | 13 | 0100 | 0 |
| 6 | 1010 | 1 | 14 | 0010 | 0 |
| 7 | 0101 | 0 | | | 0 |