A DL-Pascal 1.2 and HD61700 Series
Fascicle II – HZeta() a ζ(3) Implementation in HD61700

Table of Contents

Introduction..... 1
Chapter 7: Accessing array elements.....2
Chapter 8: HZeta().....4
Acknowledgments..... 8

Introduction

This is the second in a series of fascicles exploring direct programming of the HD61700 processor in the Casio PB-2000C. PB-2000C at The Computer Museum in Munich.

It is recommended that you read the first fascicle, since it establishes the foundation for the work presented here.

This fascicle presents an HD61700 program that computes the first three terms of ζ(3), using fixed-point arithmetic scaled by 10000 and the DL-Pascal Word data type. DL-Pascal 1.2 is used as the host environment.

This document summarizes the HD61700 and DL-Pascal techniques established during this work. It is intended as a technical reference and as background material for an AI assistant attempting to reproduce or continue these experiments.

Piotr's translation of the HD61700 manual is an invaluable resource for training humans and AI systems alike.

Chapter 7: Accessing array elements

Since the computation of the first terms of ζ(3) relies on a set of constants, we first needed a technique to make those constants available to the HD61700 program.

One possibility would have been to define the constants directly in the assembly source using DW directives and access them by name. This presents a problem when the assembled machine code is transferred into a DL-Pascal INLINE statement: we have found no documented mechanism for relocating such addresses to the location where DL-Pascal eventually places the code. We therefore decided that, for this experiment, the machine-code routine should not depend on knowing the absolute address of its data. Relocation of data references and branching instructions within INLINE code is an interesting subject in its own right, and we plan to investigate it in a later episode.

We therefore developed a technique in which DL-Pascal allocates and initializes the constants as an array of Word, and the address of its first element is passed to the HD61700 routine as a VAR parameter. The machine-code routine can then use this address as the base of the array and access individual elements by calculating their byte offsets.

Before using the technique in the ζ(3) program, we wrote the following small program to test it. HGet receives the address of dw[0] and an index, calculates the address of the corresponding array element, and returns its value.

```
PROGRAM HGet_Test;

VAR
  dw : ARRAY[0..4] OF Word;

FUNCTION HGet(VAR w : Word; idx : Word) : Word;
BEGIN
  INLINE(
    $42,$04,$00,          { LD   $4,0               }
    $A8,$68,$04,          { LDW  $8,(IX+$4)         }
    $42,$04,$02,          { LD   $4,2               }
    $A8,$6A,$04,          { LDW  $10,(IX+$4)        }
    $98,$6A,              { BIUW $10                }
    $88,$68,$0A,          { ADW  $8,$10             }
    $91,$66,$08,          { LDW  $6,($8)            }
    $42,$04,$04,          { LD   $4,4               }
    $A0,$E6,$04);         { STW  $6,(IX-$4)         }
END;

BEGIN
  dw[0] := 77;
  dw[1] := 156;
  dw[2] := 133;
  dw[3] := 1397;
  dw[4] := 648;

  WriteLn('dw[0] = ',HGet(dw[0],0));
  WriteLn('dw[1] = ',HGet(dw[0],1));
  WriteLn('dw[2] = ',HGet(dw[0],2));
  WriteLn('dw[3] = ',HGet(dw[0],3));
  WriteLn('dw[4] = ',HGet(dw[0],4));
END.
```

Getting the expected output:

```
dw[0] = 77
dw[1] = 156
dw[2] = 133
dw[3] = 1397
dw[4] = 648
```

Chapter 8: HZeta()

With the constants now accessible to the HD61700 code, all the pieces developed so far can be brought together. HZeta() performs the complete three-term ζ(3) calculation in a single HD61700 INLINE block, using the Bank 0 multiplication and division routines and the array-access technique developed in the previous chapter.

```
PROGRAM Zeta_HD61700;

VAR
  dw : ARRAY[0..9] OF Word;


FUNCTION HZeta(VAR w : Word) : Word;
BEGIN
  INLINE(

    $42,$04,$00,              { LD      $4,0                }
    $A8,$74,$04,              { LDW     $20,(IX+$4)         }

    { ------------------------------------------------- }
    { t0 = 77*156 + 77/4                                  }
    { ------------------------------------------------- }

    $82,$6A,$14,              { LDW     $10,$20             }
    $D1,$0C,$00,$00,          { LDW     $12,0               }
    $88,$6A,$0C,              { ADW     $10,$12             }
    $91,$65,$0A,              { LDW     $5,($10)            }

    $82,$6A,$14,              { LDW     $10,$20             }
    $D1,$0C,$02,$00,          { LDW     $12,2               }
    $88,$6A,$0C,              { ADW     $10,$12             }
    $91,$6F,$0A,              { LDW     $15,($10)           }

    $FF,                      { TRP                         }
    $D6,$0A,                  { DW      &H0AD6              }

    $82,$76,$00,              { LDW     $22,$0              }

    $82,$6A,$14,              { LDW     $10,$20             }
    $D1,$0C,$00,$00,          { LDW     $12,0               }
    $88,$6A,$0C,              { ADW     $10,$12             }
    $91,$6F,$0A,              { LDW     $15,($10)           }

    $82,$6A,$14,              { LDW     $10,$20             }
    $D1,$0C,$04,$00,          { LDW     $12,4               }
    $88,$6A,$0C,              { ADW     $10,$12             }
    $91,$65,$0A,              { LDW     $5,($10)            }
```

```
$FF,                    { TRP                       }
$E8,$0A,                { DW      &H0AE8            }

$88,$76,$0F,            { ADW     $22,$15           }


{ ------------------------------------------------- }
{ t1 = (((133*25)/24)*25)/324                       }
{ ------------------------------------------------- }

$82,$6A,$14,            { LDW     $10,$20           }
$D1,$0C,$06,$00,        { LDW     $12,6             }
$88,$6A,$0C,            { ADW     $10,$12           }
$91,$65,$0A,            { LDW     $5,($10)          }

$82,$6A,$14,            { LDW     $10,$20           }
$D1,$0C,$08,$00,        { LDW     $12,8             }
$88,$6A,$0C,            { ADW     $10,$12           }
$91,$6F,$0A,            { LDW     $15,($10)         }

$FF,                    { TRP                       }
$D6,$0A,                { DW      &H0AD6            }

$82,$6F,$00,            { LDW     $15,$0            }

$82,$6A,$14,            { LDW     $10,$20           }
$D1,$0C,$0A,$00,        { LDW     $12,10            }
$88,$6A,$0C,            { ADW     $10,$12           }
$91,$65,$0A,            { LDW     $5,($10)          }

$FF,                    { TRP                       }
$E8,$0A,                { DW      &H0AE8            }

$82,$65,$0F,            { LDW     $5,$15            }

$82,$6A,$14,            { LDW     $10,$20           }
$D1,$0C,$08,$00,        { LDW     $12,8             }
$88,$6A,$0C,            { ADW     $10,$12           }
$91,$6F,$0A,            { LDW     $15,($10)         }

$FF,                    { TRP                       }
$D6,$0A,                { DW      &H0AD6            }

$82,$6F,$00,            { LDW     $15,$0            }

$82,$6A,$14,            { LDW     $10,$20           }
$D1,$0C,$0C,$00,        { LDW     $12,12            }
$88,$6A,$0C,            { ADW     $10,$12           }
$91,$65,$0A,            { LDW     $5,($10)          }
```

```
    $FF,                        { TRP                           }
    $E8,$0A,                    { DW      &H0AE8                }

    $89,$76,$0F,                { SBW     $22,$15               }


    { ------------------------------------------------- }
    { t2 = (1397/240)/648                                 }
    { ------------------------------------------------- }

    $82,$6A,$14,                { LDW     $10,$20               }
    $D1,$0C,$0E,$00,            { LDW     $12,14                }
    $88,$6A,$0C,                { ADW     $10,$12               }
    $91,$6F,$0A,                { LDW     $15,($10)             }

    $82,$6A,$14,                { LDW     $10,$20               }
    $D1,$0C,$10,$00,            { LDW     $12,16                }
    $88,$6A,$0C,                { ADW     $10,$12               }
    $91,$65,$0A,                { LDW     $5,($10)              }

    $FF,                        { TRP                           }
    $E8,$0A,                    { DW      &H0AE8                }

    $82,$6A,$14,                { LDW     $10,$20               }
    $D1,$0C,$12,$00,            { LDW     $12,18                }
    $88,$6A,$0C,                { ADW     $10,$12               }
    $91,$65,$0A,                { LDW     $5,($10)              }

    $FF,                        { TRP                           }
    $E8,$0A,                    { DW      &H0AE8                }

    $88,$76,$0F,                { ADW     $22,$15               }


    { ------------------------------------------------- }
    { Return result                                       }
    { ------------------------------------------------- }

    $42,$04,$04,                { LD      $4,4                  }
    $A0,$F6,$04);               { STW     $22,(IX-$4)           }

END;


BEGIN
  dw[0] := 77;
  dw[1] := 156;
  dw[2] := 4;
  dw[3] := 133;
  dw[4] := 25;
  dw[5] := 24;
```

```
  dw[6] := 324;
  dw[7] := 1397;
  dw[8] := 240;
  dw[9] := 648;

  WriteLn('HD61700 Zeta = ',HZeta(dw[0]));
END.
```

With the expected output:

```
HD61700 Zeta = 12021
```

With just seven instructions so far, a few more to go.

```
LD    Load an 8-bit register, principally $4 for IX displacement
LDW   Load a Word: immediate, register-to-register, IX-relative, and indirect memory
STW   Store a Word, particularly the Pascal function result
ADW   16-bit Word addition; also used for address calculation
SBW   16-bit Word subtraction
BIUW  16-bit Word left shift ( shl 1 in Pascal or  << 1  in C)
TRP   Enter the ROM trap mechanism used to call Bank 0 routines
```

Acknowledgments

This document was produced by Xandrian Labs. 付喪神.

Special thanks to Piotr. His dedication to preserving vintage electronics made this possible.

This is a collaboration with ChatGPT-5.6 Sol. All errors are human.

This work is freely shared with the vintage computing community. Please use it, build upon it, and cite the source.

Revision 3.