\\ множення двох многочленів степеня 5 та 4

DATA

const m = 4 \\ значення m визначає розрядність пам’яті даних

const pow\_A = 5 \\ всі константи, окрім m записуються у пам’ять даних

const pow\_B = 4 \\ компілятор має перевіряти, щоб у константу нічого не записувалось далі

Array A [pow\_A:0] = (12, 15, 1, 0, 7, 11) \\ ліворуч молодший коефіцієнт

Array B [pow\_B:0] = (7, 1, 4, 3, 10)

Array C [pow\_A+pow\_B:0] = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

CODE

LOAD\_CA\_A CA\_2, 9b’000000000

LOOP 0, pow\_B + 1 \\ якщо кількість повторень циклу буде не константа, а число, то

\\ записати це число у пам’ять даних, а сюди вставити його адресу в пам’яті даних LOAD\_CA\_A CA\_1, 9b’000000000 \\ передбачити у компіляторі можливість задання

\\ константи у десятковому вигляді та переведення її \\ до дев’ятирозрядного двійкового вигляду

LOAD\_CA CA\_3, CA\_2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MOV\_ARRAY | R1, B [CA\_2 + 0] | | |  |
| LOOP 1, pow\_A + 1 | | |  |  |
| MOV\_ARRAY | | | R0, A [CA\_1 + 0] | \\ передбачити, щоб при відсутності зміщення |
| MOV\_ARRAY | | | R2, C [CA\_3 + 0] | \\ компілятор це сприймав як значення нуль |
| MULT | R0, R1 | |  |  |
| ADD | R2, R0 | |  |  |
| MOV\_ARRAY | | | C [CA\_3 + 0], R2 |  |
| INC\_DEC | | CA\_1, 0 | |  |
| INC\_DEC | | CA\_3, 0 | |  |
| END\_LOOP | 1 |  |  |  |

INC\_DEC CA\_2, 0

END\_LOOP 0

LOAD\_CA\_A CA\_1, 9b’000000000

LOOP 0, pow\_A + pow\_B + 1 \\ додавання виконує компілятор і заносить це число у пам’ять

\\ даних за окремою адресою на яку потім виконується \\ посилання у пам’яті команд, те саме з додаванням 1

OUT C [CA\_1 + 0]

INC\_DEC CA\_1, 0

END\_LOOP 0

1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Пам’ять даних** | |  | **Пам’ять команд** | |  |  |  |
|  | **(розрядність *m*, кількість слів 512)** | |  | **(розрядність 9, кількість слів 512)** | | | | |
| 0 | 0101 | // pow\_A | 0 | 11011000 | // LOAD\_CA\_A | | | CA\_2, 9b’000000000 |
| 1 | 0100 | // pow\_B | 1 | 00000000 |  |  |  |  |
| 2 | 0101 | // pow\_B + 1 | 2 | 10110000 | // LOOP | 0, pow\_B + 1 | | |
| 3 | 0110 | // pow\_A + 1 | 3 | 00000010 |  |  |  |  |
| 4 | 1010 | // pow\_A + pow\_B + 1 | 4 | 10100000 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 5 | 1100 | // масив А | 5 | 00100010 |  |  |  |  |
| 6 | 1111 |  | 6 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
| 7 | 0001 |  | 7 | 00000000 |  |  |  |  |
| 8 | 0000 |  | 8 | 11001110 | // LOAD\_CA | | CA\_3, CA\_2 | |
| 9 | 0111 |  | 9 | 10010110 | // MOV\_ARRAY | | | R1, B [CA\_2 + 0] |
| 10 | 1011 |  | 10 | 00001011 |  |  |  |  |
| 11 | 0111 | // масив В | 11 | 10010000 |  |  |  |  |
| 12 | 0001 |  | 12 | 10110100 | // LOOP | 1, pow\_A + 1 | | |
| 13 | 0100 |  | 13 | 00000011 |  |  |  |  |
| 14 | 0011 |  | 14 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 15 | 1010 |  | 15 | 00011111 |  |  |  |  |
| 16 | 0000 | // масив С | 16 | 10010010 | // MOV\_ARRAY | | | R0, A [CA\_1 + 0] |
| 17 | 0000 |  | 17 | 00000101 |  |  |  |  |
| 18 | 0000 |  | 18 | 01010000 |  |  |  |  |
| 19 | 0000 |  | 19 | 10011010 | // MOV\_ARRAY | | | R2, C [CA\_3 + 0] |
| 20 | 0000 |  | 20 | 00010000 |  |  |  |  |
| 21 | 0000 |  | 21 | 11010000 |  |  |  |  |
| 22 | 0000 |  | 22 | 00010001 | // MULT | R0, R1 | | |
| 23 | 0000 |  | 23 | 00001000 | // ADD | R2, R0 | |  |
| 24 | 0000 |  | 24 | 10011000 | // MOV\_ARRAY | | | C [CA\_3 + 0], R2 |
| 25 | 0000 |  | 25 | 00010000 |  |  |  |  |
|  |  |  | 26 | 11010000 |  |  |  |  |
|  |  |  | 27 | 11100110 | // INC\_DEC | | CA\_1, 0 | |
|  |  |  | 28 | 11101110 | // INC\_DEC | | CA\_3, 0 | |
|  |  |  | 29 | 10101110 | // END\_LOOP, безумовний перехід за адресою | | | |
|  |  |  | 30 | 00001110 | // перехід на початок циклу | | | |
|  |  |  | 31 | 11101010 | // INC\_DEC | | CA\_2, 0 | |
|  |  |  | 32 | 10101110 | // END\_LOOP, безумовний перехід за адресою | | | |
|  |  |  | 33 | 00000100 | // перехід на початок циклу | | | |
|  |  |  | 34 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
|  |  |  | 35 | 00000000 |  |  |  |  |
|  |  |  | 36 | 10110100 | // LOOP | 0, pow\_A + pow\_B + 1 | | |
|  |  |  | 37 | 00000100 |  |  |  |  |
|  |  |  | 38 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
|  |  |  | 39 | 00101110 |  |  |  |  |
|  |  |  | 40 | 11110010 | // OUT | C [CA\_1 + 0] | | |
|  |  |  | 41 | 00010000 |  |  |  |  |
|  |  |  | 42 | 01000000 |  |  |  |  |
|  |  |  | 43 | 11100110 | // INC\_DEC | | CA\_1, 0 | |
|  |  |  | 44 | 10101110 | // END\_LOOP, безумовний перехід за адресою | | | |
|  |  |  | 45 | 00100110 | // перехід на початок циклу | | | |

2

\\ генерування твірного многочлена степеня r

DATA

const m = 4 \\ значення m визначає розрядність пам’яті даних const r = 7

const pow\_B = 1 const pow\_T = r const alpha = 2 const sqr\_alpha = 4 const null = 0; count\_T = 2

Array G [r:0] = (alpha, 1, 0, 0, 0, 0, 0, 0) \\ ліворуч молодший коефіцієнт Array T [pow\_T:0] = (alpha, 1, 0, 0, 0, 0, 0, 0)

Array B [pow\_B:0] = (sqr\_alpha, 1)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CODE |  |  |  |  |  |  |  |  |
| MOV\_A | R0, alpha | | |  |  |  |  |  |
| LOAD\_CA\_A | | CA\_0, 9b’000000000 | | | | | |  |
| LOOP | 0, r – 1 | |  |  |  |  |  |  |
|  | LOAD\_CA\_A | | | CA\_2, 9b’000000000 | | | | |
|  | LOOP | 1, pow\_B + 1 | | | | |  |  |
|  |  | LOAD\_CA\_A | | | | CA\_1, 9b’000000000 | | |
|  |  | LOAD\_CA | | | CA\_3, CA\_2 | | | |
|  |  | MOV\_ARRAY | | | | R2, B [CA\_2 + 0] | | |
|  |  | LOOP | | 2, count\_T | | | |  |
|  |  |  |  | MOV\_ARRAY | | | | R1, T [CA\_1 + 0] |
|  |  |  |  | MOV\_ARRAY | | | | R3, G [CA\_3 + 0] |
|  |  |  |  | MULT | | R1, R2 | |  |
|  |  |  |  | ADD | | R3, R1 | |  |
|  |  |  |  | MOV\_ARRAY | | | | G [CA\_3 + 0], R3 |
|  |  |  |  | INC\_DEC | | | CA\_1, 0 | |
|  |  |  |  | INC\_DEC | | | CA\_3, 0 | |
|  |  | END\_LOOP | | |  | 2 |  |  |
|  |  | INC\_DEC\_CA | | | | CA\_2, 0 | |  |
|  | END\_LOOP | | | 1 |  |  |  |  |
|  | MOV\_ARRAY | | | R1, B [CA\_0 + 0] | | | | |
|  | MULT | R1, R0 | | |  |  |  |  |
|  | MOV\_ARRAY | | | B [CA\_0 + 0], R1 | | | | |
|  | MOV\_A | | R1, count\_T | | | |  |  |
|  | INC\_DEC | | R1, 0 | |  |  |  |  |
|  | MOV\_A | | count\_T, R1 | | | |  |  |
|  | MOV\_A | | R2, null | |  |  |  |  |
|  | LOAD\_CA\_A | | | CA\_1, 9b’000000000 | | | | |
|  | LOOP | 1, count\_T | | | |  |  |  |
|  |  | MOV\_ARRAY | | | | R1, G [CA\_1 + 0] | | |
|  |  | MOV\_ARRAY | | | | T [CA\_1 + 0], R1 | | |
|  |  | MOV\_ARRAY | | | | G [CA\_1 + 0], R2 | | |
|  |  | INC\_DEC | | | CA\_1, 0 | |  |  |
|  | END\_LOOP | | | 1 |  |  |  |  |
| END\_LOOP | | 0 |  |  |  |  |  |  |
| LOAD\_CA\_A | | CA\_1, 9b’000000000 | | | | | |  |
| LOOP | 0, r + 1 | |  |  |  |  |  |  |
|  | OUT | G [CA\_1 + 0] | | | |  |  |  |
|  | INC\_DEC | | CA\_1, 0 | |  |  |  |  |
| END\_LOOP | | 0 |  |  |  |  |  |  |

3

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Пам’ять даних** | |  | **Пам’ять команд** | |  |  |  |
|  | **(розрядність *m*, кількість слів 512)** | |  | **(розрядність 9, кількість слів 512)** | | | | |
| 0 | 0111 | // r | 0 | 10000010 // MOV\_A | |  | R0, alpha | |
| 1 | 0001 | // pow\_B | 1 | 00000100 |  |  |  |  |
| 2 | 0111 | // pow\_T | 2 | 11010000 | // LOAD\_CA\_A | | | CA\_0, 9b’000000000 |
| 3 | 0010 | // count\_T | 3 | 00000000 |  |  |  |  |
| 4 | 0010 | // alpha | 4 | 10110000 | // LOOP | 0, r – 1 | | |
| 5 | 0000 | // null | 5 | 00000111 |  |  |  |  |
| 6 | 0010 | // pow\_B + 1 | 6 | 10100000 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 7 | 0110 | // r – 1 | 7 | 01001100 |  |  |  |  |
| 8 | 0010 | // масив G | 8 | 10001010 | // MOV\_A |  | R2, null | |
| 9 | 0001 |  | 9 | 00000101 |  |  |  |  |
| 10 | 0000 |  | 10 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
| 11 | 0000 |  | 11 | 00000000 |  |  |  |  |
| 12 | 0000 |  | 12 | 10110100 | // LOOP | 1, count\_T | | |
| 13 | 0000 |  | 13 | 00000011 |  |  |  |  |
| 14 | 0000 |  | 14 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 15 | 0000 |  | 15 | 00011100 |  |  |  |  |
| 16 | 0010 | // масив T | 16 | 10010110 | // MOV\_ARRAY | | | R1, G [CA\_1 + 0] |
| 17 | 0001 |  | 17 | 00001000 |  |  |  |  |
| 18 | 0000 |  | 18 | 01000000 |  |  |  |  |
| 19 | 0000 |  | 19 | 10010100 | // MOV\_ARRAY | | | T [CA\_1 + 0], R1 |
| 20 | 0000 |  | 20 | 00010000 |  |  |  |  |
| 21 | 0000 |  | 21 | 01000000 |  |  |  |  |
| 22 | 0000 |  | 22 | 10011000 | // MOV\_ARRAY | | | G [CA\_1 + 0], R2 |
| 23 | 0000 |  | 23 | 00001000 |  |  |  |  |
| 24 | 0100 | // масив В | 24 | 01000000 |  |  |  |  |
| 25 | 0001 |  | 25 | 11100110 | // INC\_DEC | | CA\_1, 0 | |
|  |  |  | 26 | 10101110 | // END\_LOOP, безумовний перехід за адресою | | | |
|  |  |  | 27 | 00001110 | // перехід на початок циклу | | | |
|  |  |  | 28 | 11011000 | // LOAD\_CA\_A | | | CA\_2, 9b’000000000 |
|  |  |  | 29 | 00000000 |  |  |  |  |
|  |  |  | 30 | 10110100 | // LOOP | 1, pow\_B + 1 | | |
|  |  |  | 31 | 00000110 |  |  |  |  |
|  |  |  | 32 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
|  |  |  | 33 | 00111110 |  |  |  |  |
|  |  |  | 34 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
|  |  |  | 35 | 00000000 |  |  |  |  |
|  |  |  | 36 | 11001110 | // LOAD\_CA | | CA\_3, CA\_2 | |
|  |  |  | 37 | 10011010 | // MOV\_ARRAY | | | R2, B [CA\_2 + 0] |
|  |  |  | 38 | 00011000 |  |  |  |  |
|  |  |  | 39 | 10010000 |  |  |  |  |
|  |  |  | 40 | 10111000 | // LOOP | 2, count\_T | | |
|  |  |  | 41 | 00000011 |  |  |  |  |
|  |  |  | 42 | 10101000 | // якщо лічильник циклу = нуль, то виходимо | | | |
|  |  |  | 43 | 00111011 |  |  |  |  |
|  |  |  | 44 | 10010110 | // MOV\_ARRAY | | | R1, T [CA\_1 + 0] |
|  |  |  | 45 | 00010000 |  |  |  |  |
|  |  |  | 46 | 01010000 |  |  |  |  |
|  |  |  | 47 | 10011110 | // MOV\_ARRAY | | | R3, G [CA\_3 + 0] |
|  |  |  | 48 | 00001000 |  |  |  |  |
|  |  |  | 49 | 11010000 |  |  |  |  |
|  |  |  | 50 | 00010110 | // MULT | R1, R2 | | |

4

|  |  |  |
| --- | --- | --- |
| 51 | 00001101 | // ADD R3, R1 |
| 52 | 10011100 | // MOV\_ARRAY G [CA\_3 + 0], R3 |

1. 00001000
2. 11010000

|  |  |  |
| --- | --- | --- |
| 55 | 11100110 // INC\_DEC | CA\_1, 0 |
| 56 | 11101110 // INC\_DEC | CA\_3, 0 |

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 00101010 // перехід на початок циклу

59 11101010 // INC\_DEC CA\_2, 0

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 00100000 // перехід на початок циклу

62 10010110 // MOV\_ARRAY R1, B [CA\_0 + 0]

1. 00011000
2. 00010000

|  |  |  |
| --- | --- | --- |
| 65 | 00010100 | // MULT R1, R0 |
| 66 | 10010100 | // MOV\_ARRAY B [CA\_0 + 0], R1 |

1. 00011000
2. 00010000

|  |  |  |
| --- | --- | --- |
| 69 | 10000110 // MOV\_A | R1, count\_T |
| 70 | 00000011 |  |
| 71 | 11100100 // INC\_DEC | R1, 0 |
| 72 | 10000100 // MOV\_A | count\_T, R1 |

1. 00000011
2. 10101110 // END\_LOOP, безумовний перехід за адресою
3. 00000110 // перехід на початок циклу

|  |  |  |
| --- | --- | --- |
| 76 | 11010100 | // LOAD\_CA\_A CA\_1, 9b’000000000 |
| 77 | 00000000 |  |
| 78 | 10110000 | // LOOP 0, count\_T |

1. 00000011
2. 10100000 // якщо лічильник циклу = нуль, то виходимо
3. 01011000

82 11110010 // OUT G [CA\_1 + 0]

1. 00001000
2. 01000000

85 11100110 // INC\_DEC CA\_1, 0

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 01010000 // перехід на початок циклу

5

\\ Кодування кодом Ріда-Соломона

DATA

const m = 4 \\ значення m визначає розрядність пам’яті даних const r = 7

const pow\_B = 1 const pow\_T = r const alpha = 2 const sqr\_alpha = 4 const null = 0; const pow\_A = 5 count\_T = 2

Array G [r:0] = (alpha, 1, 0, 0, 0, 0, 0, 0) \\ ліворуч молодший коефіцієнт Array T [pow\_T:0] = (alpha, 1, 0, 0, 0, 0, 0, 0)

Array B [pow\_B:0] = (sqr\_alpha, 1) Array A [pow\_A:0] = (5, 15, 1, 0, 7, 3)

Array C [pow\_A + r : 0] = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

CODE

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MOV\_A | R0, alpha | | |  |  |  |  |  |
| LOAD\_CA\_A | | CA\_0, 9b’000000000 | | | | | |  |
| LOOP | 0, r – 1 | |  |  |  |  |  |  |
|  | LOAD\_CA\_A | | | CA\_2, 9b’000000000 | | | | |
|  | LOOP |  | 1, pow\_B + 1 | | | |  |  |
|  |  |  | LOAD\_CA\_A | | | CA\_1, 9b’000000000 | | |
|  |  |  | LOAD\_CA | | CA\_3, CA\_2 | | | |
|  |  |  | MOV\_ARRAY | | | R2, B [CA\_2 + 0] | | |
|  |  |  | LOOP | 2, count\_T | | | |  |
|  |  |  |  | MOV\_ARRAY | | | | R1, T [CA\_1 + 0] |
|  |  |  |  | MOV\_ARRAY | | | | R3, G [CA\_3 + 0] |
|  |  |  |  | MULT | | R1, R2 | |  |
|  |  |  |  | ADD | | R3, R1 | |  |
|  |  |  |  | MOV\_ARRAY | | | | G [CA\_3 + 0], R3 |
|  |  |  |  | INC\_DEC | | | CA\_1, 0 | |
|  |  |  |  | INC\_DEC | | | CA\_3, 0 | |
|  |  |  | END\_LOOP | |  | 2 |  |  |
|  |  |  | INC\_DEC\_CA | | | CA\_2, 0 | |  |
|  | END\_LOOP | | | 1 |  |  |  |  |
|  | MOV\_ARRAY | | | R1, B [CA\_0 + 0] | | | | |
|  | MULT |  | R1, R0 | |  |  |  |  |
|  | MOV\_ARRAY | | | B [CA\_0 + 0], R1 | | | | |
|  | MOV\_A | | R1, count\_T | | | |  |  |
|  | INC\_DEC R1, 0 | | | |  |  |  |  |
|  | MOV\_A | | count\_T, R1 | | | |  |  |
|  | MOV\_A | | R2, null | |  |  |  |  |
|  | LOAD\_CA\_A | | | CA\_1, 9b’000000000 | | | | |
|  | LOOP |  | 1, count\_T | | |  |  |  |
|  |  |  | MOV\_ARRAY | | | R1, G [CA\_1 + 0] | | |
|  |  |  | MOV\_ARRAY | | | T [CA\_1 + 0], R1 | | |
|  |  |  | MOV\_ARRAY | | | G [CA\_1 + 0], R2 | | |
|  |  |  | INC\_DEC | | CA\_1, 0 | |  |  |
|  | END\_LOOP | | | 1 |  |  |  |  |
| END\_LOOP | | 0 |  |  |  |  |  |  |

6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| LOAD\_CA\_A | | CA\_2, 9b’000000000 // множення твірного многочлена на многочлен даних | | | | |
| LOOP | 0, pow\_A + 1 | | | |  |  |
|  | LOAD\_CA\_A | | | CA\_1, 9b’000000000 | | |
|  | LOAD\_CA | | CA\_3, CA\_2 | | | |
|  | MOV\_ARRAY | | | R1, A [CA\_2 + 0] | | |
|  | LOOP | 1, count\_T | | | |  |
|  |  | MOV\_ARRAY | | | | R0, G [CA\_1 + 0] |
|  |  | MOV\_ARRAY | | | | R2, C [CA\_3 + 0] |
|  |  | MULT | | R0, R1 | |  |
|  |  | ADD | | R2, R0 | |  |
|  |  | MOV\_ARRAY | | | | C [CA\_3 + 0], R2 |
|  |  | INC\_DEC | | | CA\_1, 0 | |
|  |  | INC\_DEC | | | CA\_3, 0 | |
|  | END\_LOOP | |  | 1 |  |  |
|  | INC\_DEC | | CA\_2, 0 | |  |  |
| END\_LOOP | | 0 |  |  |  |  |
| LOAD\_CA\_A | | CA\_1, 9b’000000000 | | | | |
| LOOP | 0, pow\_A + r + 1 | | | |  |  |
|  | OUT | C [CA\_1 + 0] | | |  |  |
|  | INC\_DEC | | CA\_1, 0 | |  |  |
| END\_LOOP | | 0 |  |  |  |  |

7

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Пам’ять даних** | |  | **Пам’ять команд** | |  |  |  |
|  | **(розрядність *m*, кількість слів 512)** | |  | **(розрядність 9, кількість слів 512)** | | | | |
| 0 | 0111 | // r | 0 | 10000010 // MOV\_A | |  | R0, alpha | |
| 1 | 0001 | // pow\_B | 1 | 00000100 |  |  |  |  |
| 2 | 0111 | // pow\_T | 2 | 11010000 | // LOAD\_CA\_A | | | CA\_0, 9b’000000000 |
| 3 | 0010 | // count\_T | 3 | 00000000 |  |  |  |  |
| 4 | 0010 | // alpha | 4 | 10110000 | // LOOP | 0, r – 1 | | |
| 5 | 0000 | // null | 5 | 00000111 |  |  |  |  |
| 6 | 0010 | // pow\_B + 1 | 6 | 10100000 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 7 | 0110 | // r – 1 | 7 | 01001100 |  |  |  |  |
| 8 | 0101 | // pow\_A | 8 | 10001010 // MOV\_A | |  | R2, null | |
| 9 | 0110 | // pow\_A + 1 | 9 | 00000101 |  |  |  |  |
| 10 | 1101 | // pow\_A + r + 1 | 10 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
| 11 | 0010 | // масив G | 11 | 00000000 |  |  |  |  |
| 12 | 0001 |  | 12 | 10110100 | // LOOP | 1, count\_T | | |
| 13 | 0000 |  | 13 | 00000011 |  |  |  |  |
| 14 | 0000 |  | 14 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 15 | 0000 |  | 15 | 00011100 |  |  |  |  |
| 16 | 0000 |  | 16 | 10010110 | // MOV\_ARRAY | | | R1, G [CA\_1 + 0] |
| 17 | 0000 |  | 17 | 00001011 |  |  |  |  |
| 18 | 0000 |  | 18 | 01000000 |  |  |  |  |
| 19 | 0010 | // масив T | 19 | 10010100 | // MOV\_ARRAY | | | T [CA\_1 + 0], R1 |
| 20 | 0001 |  | 20 | 00010011 |  |  |  |  |
| 21 | 0000 |  | 21 | 01000000 |  |  |  |  |
| 22 | 0000 |  | 22 | 10011000 | // MOV\_ARRAY | | | G [CA\_1 + 0], R2 |
| 23 | 0000 |  | 23 | 00001011 |  |  |  |  |
| 24 | 0000 |  | 24 | 01000000 |  |  |  |  |
| 25 | 0000 |  | 25 | 11100110 | // INC\_DEC | | CA\_1, 0 | |
| 26 | 0000 |  | 26 | 10101110 | // END\_LOOP, безумовний перехід за адресою | | | |
| 27 | 0100 | // масив В | 27 | 00001110 | // перехід на початок циклу | | | |
| 28 | 0001 |  | 28 | 11011000 | // LOAD\_CA\_A | | | CA\_2, 9b’000000000 |
| 29 | 0101 | // масив A | 29 | 00000000 |  |  |  |  |
| 30 | 1111 |  | 30 | 10110100 | // LOOP | 1, pow\_B + 1 | | |
| 31 | 0001 |  | 31 | 00000110 |  |  |  |  |
| 32 | 0000 |  | 32 | 10100100 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 33 | 0111 |  | 33 | 00111110 |  |  |  |  |
| 34 | 0011 |  | 34 | 11010100 | // LOAD\_CA\_A | | | CA\_1, 9b’000000000 |
| 35 | 0000 | // масив C | 35 | 00000000 |  |  |  |  |
| 36 | 0000 |  | 36 | 11001110 | // LOAD\_CA | | CA\_3, CA\_2 | |
| 37 | 0000 |  | 37 | 10011010 | // MOV\_ARRAY | | | R2, B [CA\_2 + 0] |
| 38 | 0000 |  | 38 | 00011011 |  |  |  |  |
| 39 | 0000 |  | 39 | 10010000 |  |  |  |  |
| 40 | 0000 |  | 40 | 10111000 | // LOOP | 2, count\_T | | |
| 41 | 0000 |  | 41 | 00000011 |  |  |  |  |
| 42 | 0000 |  | 42 | 10101000 | // якщо лічильник циклу = нуль, то виходимо | | | |
| 43 | 0000 |  | 43 | 00111011 |  |  |  |  |
| 44 | 0000 |  | 44 | 10010110 | // MOV\_ARRAY | | | R1, T [CA\_1 + 0] |
| 45 | 0000 |  | 45 | 00010011 |  |  |  |  |
| 46 | 0000 |  | 46 | 01010000 |  |  |  |  |
| 47 | 0000 |  | 47 | 10011110 | // MOV\_ARRAY | | | R3, G [CA\_3 + 0] |
|  |  |  | 48 | 00001011 |  |  |  |  |
|  |  |  | 49 | 11010000 |  |  |  |  |

8

|  |  |  |  |
| --- | --- | --- | --- |
| 50 | 00010110 | // MULT | R1, R2 |
| 51 | 00001101 | // ADD | R3, R1 |
| 52 | 10011100 | // MOV\_ARRAY G [CA\_3 + 0], R3 | |

1. 00001011
2. 11010000

|  |  |  |
| --- | --- | --- |
| 55 | 11100110 // INC\_DEC | CA\_1, 0 |
| 56 | 11101110 // INC\_DEC | CA\_3, 0 |

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 00101010 // перехід на початок циклу

59 11101010 // INC\_DEC CA\_2, 0

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 00100000 // перехід на початок циклу

62 10010110 // MOV\_ARRAY R1, B [CA\_0 + 0]

1. 00011011
2. 00010000

|  |  |  |
| --- | --- | --- |
| 65 | 00010100 | // MULT R1, R0 |
| 66 | 10010100 | // MOV\_ARRAY B [CA\_0 + 0], R1 |

1. 00011011
2. 00010000

|  |  |  |
| --- | --- | --- |
| 69 | 10000110 // MOV\_A | R1, count\_T |
| 70 | 00000011 |  |
| 71 | 11100100 // INC\_DEC | R1, 0 |
| 72 | 10000100 // MOV\_A | count\_T, R1 |

1. 00000011
2. 10101110 // END\_LOOP, безумовний перехід за адресою
3. 00000110 // перехід на початок циклу

|  |  |  |
| --- | --- | --- |
| 76 | 11011000 | // LOAD\_CA\_A CA\_2, 9b’000000000 |
| 77 | 00000000 |  |
| 78 | 10110000 | // LOOP 0, pow\_A + 1 |

1. 00001001
2. 10100000 // якщо лічильник циклу = нуль, то виходимо
3. 01101110

|  |  |  |  |
| --- | --- | --- | --- |
| 82 | 11010100 | // LOAD\_CA\_A | CA\_1, 9b’000000000 |
| 83 | 00000000 |  |  |
| 84 | 11001110 | // LOAD\_CA | CA\_3, CA\_2 |
| 85 | 10010110 | // MOV\_ARRAY R1, A [CA\_2 + 0] | |

1. 00011101
2. 10010000

88 10110100 // LOOP 1, count\_T

1. 00000011
2. 10100100 // якщо лічильник циклу = нуль, то виходимо
3. 01101011

92 10010010 // MOV\_ARRAY R0, G [CA\_1 + 0]

1. 00001011
2. 01010000

95 10011010 // MOV\_ARRAY R2, C [CA\_3 + 0]

1. 00100011
2. 11010000

|  |  |  |  |
| --- | --- | --- | --- |
| 98 | 00010001 // MULT | | R0, R1 |
| 99 | 00001000 | // ADD | R2, R0 |
| 100 | 10011000 | // MOV\_ARRAY C [CA\_3 + 0], R2 | |

1. 00100011
2. 11010000

|  |  |  |
| --- | --- | --- |
| 103 | 11100110 // INC\_DEC | CA\_1, 0 |
| 104 | 11101110 // INC\_DEC | CA\_3, 0 |

9

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 01011010 // перехід на початок циклу

107 11101010 // INC\_DEC CA\_2, 0

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 01010000 // перехід на початок циклу

|  |  |  |
| --- | --- | --- |
| 110 | 11010100 | // LOAD\_CA\_A CA\_1, 9b’000000000 |
| 111 | 00000000 |  |
| 112 | 10110100 | // LOOP 0, pow\_A + r + 1 |

1. 00001010
2. 10100100 // якщо лічильник циклу = нуль, то виходимо
3. 01111010

116 11110010 // OUT C [CA\_1 + 0]

1. 00100011
2. 01000000

119 11100110 // INC\_DEC CA\_1, 0

1. 10101110 // END\_LOOP, безумовний перехід за адресою
2. 01110010 // перехід на початок циклу

10

**Перелік команд:**

|  |  |
| --- | --- |
| **Команда** | **Код операції** |
| ADD\_SUB | 0000 |
| MULT | 0001 |
| DIV | 0010 |
| POW | 0011 |
| INV\_ | 0100 |
| CDP | 0101 |
| CPD | 0110 |
| MOV | 0111 |
| MOV\_A | 1000 |
| MOV\_ARRAY | 1001 |
| JMP | 1010 |
| LOOP | 1011 |
| LOAD\_CA | 1100 |
| LOAD\_CA\_A | 1101 |
| INC\_DEC | 1110 |
| OUT | 1111 |

**Команда ADD\_SUB** (однобайтна)

Додає або віднімає (це одне й те саме) значення у двох регістрах, результат записує у регістр який стоїть першим

|  |  |  |
| --- | --- | --- |
| ADD\_SUB | R0, R1 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (0000) |
| 3 – 2 |  | Номер першого регістра (регістра який є першим операндом та в який буде |
|  |  | записано результат виконання операції) |
| 1 – 0 |  | Номер другого регістра (регістра який є другим операндом) |

**Команда MULT** (однобайтна)

Множить значення у двох регістрах, результат записує у регістр який стоїть першим

|  |  |  |
| --- | --- | --- |
| MULT | R0, R1 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (0001) |
| 3 – 2 |  | Номер першого регістра (регістра який є першим операндом та в який буде |
|  |  | записано результат виконання операції) |
| 1 – 0 |  | Номер другого регістра (регістра який є другим операндом) |

11

**Команда DIV** (однобайтна)

Ділить значення з першого регістру на значення з другого регістру, результат записує у регістр який стоїть першим

DIV R0, R1

|  |  |
| --- | --- |
| Біти | Опис |
| 7 – 4 | Код операції (0010) |
| 3 – 2 | Номер першого регістра (регістра який є першим операндом та в який буде |
|  | записано результат виконання операції) |
| 1 – 0 | Номер другого регістра (регістра який є другим операндом) |

**Команда POW** (однобайтна)

Підносить до степеня (показник степеня у другому регістрі) значення з першого регістру, результат записує у регістр який стоїть першим

|  |  |  |
| --- | --- | --- |
| POW | R0, R1 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (0011) |
| 3 – 2 |  | Номер першого регістра (регістра який є першим операндом та в який буде |
|  |  | записано результат виконання операції) |
| 1 – 0 |  | Номер другого регістра (регістра який є другим операндом) |

**Команда INV\_** (однобайтна)

Шукає мультиплікативно обернений елемент до значення, що знаходиться у регістрі, результат записує у цей же регістр

|  |  |  |
| --- | --- | --- |
| INV\_ | R0 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (0100) |
| 3 – 2 |  | Номер регістра (регістр який є операндом та в який буде записано результат |
|  |  | виконання операції) |
| 1 – 0 |  | Не використовуються |

12

**Команда CDP** (однобайтна)

Переведення значення, що знаходиться у регістрі з многочленного подання у степеневе, результат записується у той самий регістр

CDP R0

|  |  |
| --- | --- |
| Біти | Опис |
| 7 – 4 | Код операції (0101) |
| 3 – 2 | Номер регістра (регістр який є операндом та в який буде записано результат |
|  | виконання операції) |
| 1 – 0 | Не використовуються |

**Команда CPD** (однобайтна)

Переведення значення, що знаходиться у регістрі зі степеневого подання у многочленне, результат записується у той самий регістр

CPD R0

|  |  |
| --- | --- |
| Біти | Опис |
| 7 – 4 | Код операції (0110) |
| 3 – 2 | Номер регістра (регістр який є операндом та в який буде записано результат |
|  | виконання операції) |
| 1 – 0 | Не використовуються |

**Команда MOV** (однобайтна)

Записує значення, що знаходиться у другому регістрі у перший регістр

|  |  |  |
| --- | --- | --- |
| MOV | R0, R1 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (0111) |
| 3 – 2 |  | Номер першого регістра |
| 1 – 0 |  | Номер другого регістра |

13

**Команда MOV\_A** (двобайтна)

Записує значення з регістру до пам’яті даних і навпаки

MOV\_A R0, A

де A – адреса у пам’яті даних з якої потрібно скопіювати дані у відповідний регістр, адреса може задаватись явно, може як змінна (оброблюється компілятором)

MOV\_A A, R0

де A – адреса у пам’яті даних до якої потрібно скопіювати дані з відповідного регістру, адреса може задаватись явно, може як змінна (оброблюється компілятором)

|  |  |  |  |
| --- | --- | --- | --- |
| Байт | Біти | Опис |  |
|  | 7 – 4 | Код операції (1000) |  |
|  | 3 – 2 | Номер регістра |  |
| 1 | 1 | Вказує на те з пам’яті даних у регістр пишемо чи з регістру у пам’ять даних (якщо |  |
|  | 1, то виконуємо запис з пам'яті в регістр; якщо 0, то виконуємо запис з регістру в |  |
|  |  |  |
|  |  | пам'ять) |  |
|  | 0 | Старший біт адреси у пам’яті даних |  |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті даних |  |

**Команда MOV\_ ARRAY** (трибайтна)

Записує значення з регістру до пам’яті даних і навпаки, використовується для роботи з масивами

MOV\_A R0, A [CA\_1 + 3]

де A – ім’я масиву, адреса у пам’яті даних з якої потрібно скопіювати дані у відповідний регістр, адреса формується як початкова адреса масиву + значення у лічильнику адрес (CA\_0, CA\_1, …, CA\_3) плюс/мінус зміщення яке задається явно

MOV\_A A [CA\_0 - 12], R2

де A – адреса у пам’яті даних до якої потрібно скопіювати дані з відповідного регістру, адреса формується як початкова адреса масиву + значення у лічильнику адрес (CA\_0, CA\_1, …, CA\_3) плюс/мінус зміщення яке задається явно

|  |  |  |  |
| --- | --- | --- | --- |
| Байт | Біти | Опис |  |
|  | 7 – 4 | Код операції (1001) |  |
|  | 3 – 2 | Номер регістра |  |
| 1 | 1 | Вказує на те з пам’яті даних у регістр пишемо чи з регістру у пам’ять даних (якщо |  |
|  | 1, то виконуємо запис з пам'яті в регістр; якщо 0, то виконуємо запис з регістру в |  |
|  |  |  |
|  |  | пам'ять) |  |
|  | 0 | Старший біт адреси у пам’яті даних (початкова адреса масиву) |  |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті даних (початкова адреса масиву) |  |
|  | 7 – 6 | Номер лічильника адрес |  |
| 3 | 5 | Не використовується |  |
| 4 | Знак зміщення |  |
|  |  |
|  | 3 – 0 | Значення зміщення |  |

14

**Команда JMP** (двобайтна)

Умовний та безумовний перехід за адресою

JMP A \\ безумовний перехід

де A – адреса пам’яті команд до якої потрібно перейти, адреса може задаватись явно, може як змінна (оброблюється компілятором)

|  |  |  |  |
| --- | --- | --- | --- |
| Байт | Біти | Опис |  |
|  | 7 – 4 | Код операції (1010) |  |
| 1 | 3 – 2 | На те що це безумовний перехід вказує значення 11 в цих бітах |  |
| 1 | На те що це безумовний перехід вказує значення 1 у цьому біті |  |
|  |  |
|  | 0 | Старший біт адреси у пам’яті команд |  |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті команд |  |

JMP R1, A \\ умовний перехід, якщо значення у відповідному регістрі дорівнює нулю, то виконуємо перехід

де A – адреса пам’яті команд до якої потрібно перейти, адреса може задаватись явно, може як змінна (оброблюється компілятором)

|  |  |  |
| --- | --- | --- |
| Байт | Біти | Опис |
|  | 7 – 4 | Код операції (1010) |
|  | 3 – 2 | Номер регістру (можна використовувати тільки 00, 01 та 10, тобто регістр R3 не |
|  |  | можна використовувати, оскільки комбінація 11 зарезервована для безумовного |
| 1 |  | переходу) |
|  | 1 | На те що це умовний перехід за нульовим значенням регістру вказує значення 1 у |
|  |  | цьому біті |
|  | 0 | Старший біт адреси у пам’яті команд |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті команд |

Окрім цього команда JMP неявним чином використовується при роботі з циклами (при формуванні машинного коду компілятором додається ця команда до машинного коду)

|  |  |  |
| --- | --- | --- |
| Байт | Біти | Опис |
|  | 7 – 4 | Код операції (1010) |
|  | 3 – 2 | Номер лічильника циклу |
| 1 | 1 | На те що це умовний перехід за нульовим значенням лічильника циклу вказує |
|  |  | значення 0 у цьому біті |
|  | 0 | Старший біт адреси у пам’яті команд |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті команд |

15

**Команда LOOP** (двобайтна)

Цикл з лічильником

LOOP 1, A \\ задаємо номер циклу (номер лічильника, який буде використовуватись) та кількість повторень циклу (значення, яке буде записано у лічильник циклу)

де A – адреса у пам’яті даних за якою записано кількість повторень циклу, адреса може задаватись явно, може як змінна (оброблюється компілятором)

|  |  |  |  |
| --- | --- | --- | --- |
| Байт | Біти | Опис |  |
|  | 7 – 4 | Код операції (1011) |  |
| 1 | 3 – 2 | Номер лічильника циклу (перший операнд у команді) |  |
| 1 | Не використовується |  |
|  |  |
|  | 0 | Старший біт адреси у пам’яті даних, де знаходиться кількість повторень циклу |  |
| 2 | 7 – 0 | Молодші біти адреси у пам’яті даних, де знаходиться кількість повторень циклу |  |

Після цієї команди одразу має йти двобайтна команда JMP умовного переходу за адресою після END\_LOOP, якщо значення відповідного лічильника циклу дорівнює нулю (додається неявно компілятором)

\\ тіло циклу

END\_LOOP 1 \\ кінець циклу (обов’язково необхідно вказати номер циклу), ця команда при перетворенні у машинний код замінюється на команду JMP безумовного переходу на команду JMP умовного переходу, що розміщена після команди LOOP

P.S. Декремент лічильника циклу виконується неявно кожного разу коли доходимо до команди END\_LOOP. Оскільки під номер лічильника циклу виділено лише 2 розряди, то можна

використовувати не більше 4 лічильників циклу, що обумовлює обмеження на кількість вкладених циклів не більше 4 у програмі. Хоч в цій команді є вільний біт і наче можна було б використати його для збільшення кількості лічильників циклу, але у команді умовного переходу немає такої можливості.

**Команда LOAD\_CA** (однобайтна)

Записує значення з другого лічильника адрес у перший (лічильник адрес використовується в якості індекса при роботі з масивом)

|  |  |  |
| --- | --- | --- |
| LOAD\_CA | CA\_1, CA\_5 | |
|  |  |  |
| Біти |  | Опис |
| 7 – 4 |  | Код операції (1100) |
| 3 – 2 |  | Номер лічильника адрес, що є першим операндом (лічильник-отримувач) |
| 1 – 0 |  | Номер лічильника адрес, що є другим операндом (лічильник-джерело) |

16

**Команда LOAD\_CA\_A** (двобайтна)

Записує значення другого операнда у відповідний лічильник адрес (лічильник адрес використовується в якості індекса при роботі з масивом)

LOAD\_CA\_A CA\_3, A

де A – адреса, яка має задаватись явно

|  |  |  |  |
| --- | --- | --- | --- |
| Байт | Біти | Опис |  |
|  | 7 – 4 | Код операції (1101) |  |
| 1 | 3 – 2 | Номер лічильника адрес |  |
| 1 | Не використовується |  |
|  |  |
|  | 0 | Старший біт адреси, що записується у лічильник адрес |  |
| 2 | 7 – 0 | Молодші біти адреси, що записується у лічильник адрес |  |

**Команда INC\_DEC** (однобайтна)

Інкремент та декремент відповідного регістра або лічильника адрес (лічильник адрес використовується в якості індекса при роботі з масивом)

INC\_DEC CA\_3, 0 \\ значення другого операнду 0 – інкремент, 1 – декремент

де A – адреса, яка має задаватись явно

|  |  |
| --- | --- |
| Біти | Опис |
| 7 – 4 | Код операції (1110) |
| 3 – 2 | Номер лічильника адрес |
| 1 | Вказує з чим працюємо: з регістром чи лічильником адрес (0 – регістр, 1 – |
|  | лічильник адрес) |
| 0 | Вказує інкремент чи декремент маємо виконувати (0 – інкремент, 1 – декремент) |

**Команда OUT** (двобайтна або трибайтна)

Виводить на вихід значення, що знаходиться за вказаною адресою у пам’яті даних

OUT A \\ приклад двобайтної команди

де A – адреса пам’яті даних з якої необхідно взяти дані, адреса може задаватись явно, може як змінна (оброблюється компілятором)

OUT B [CA\_3 + 1] \\ приклад трибайтної команди

|  |  |  |
| --- | --- | --- |
| Байт | Біти | Опис |
|  | 7 – 4 | Код операції (1111) |
|  | 3 – 2 | Не використовуються |
| 1 | 1 | Вказує двобайтна чи трибайтна команда (якщо 1, то трибайтна; якщо 0, то |
|  |  | двобайтна) |
|  | 0 | Старший біт адреси (для трибайтної команди початкова адреса масиву) |
| 2 | 7 – 0 | Молодші біти адреси (для трибайтної команди початкова адреса масиву) |
| 3 | 7 – 6 | Номер лічильника адрес |
|  | 5 | Не використовується |
|  | 4 | Знак зміщення |
|  | 3 – 0 | Значення зміщення |

17