Віртуальна машина СМ-1
(Довідник користувача)

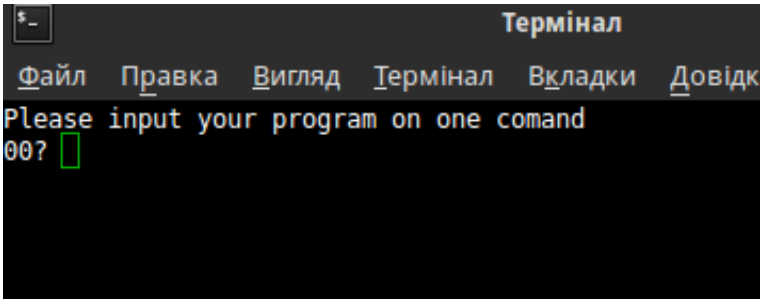
Віртуальна машина СМ-1 це досить спрощена модель комп'ютерної архітектури , яка містить у собі : процесор , представлений регістрами(акумулятор(acumulator), регістр-лічильник(count register),регістр вказівника стеку(stack pointer),регістр лічильника команд(instruction counter),регістр команд(instruction register).), оперативну пам'ять , яка містить 256 4-байтових комірок пам'яті, та апаратним стеком , що містить 64 4-байтових комірки пам'яті.
Всі команди для даної віртуальної машини(симулятора комп'ютерної архітектури) подаються в такому вигляді:

XXYY

, де XX-числова константа в 16-ричній системі числення що відповідає певній команді .
YY -числове значення в 16-ричній системі числення , що відповідає адресі , за якою потрібно здійснити дію.
Якщо ви запускаєте симулятор безпосередньо(без файла, в якому записані команди), то програма запропонує вам ввести дані ,або команди по одній починаючи з поточної адреси.
Поточна адреса має такий формат :

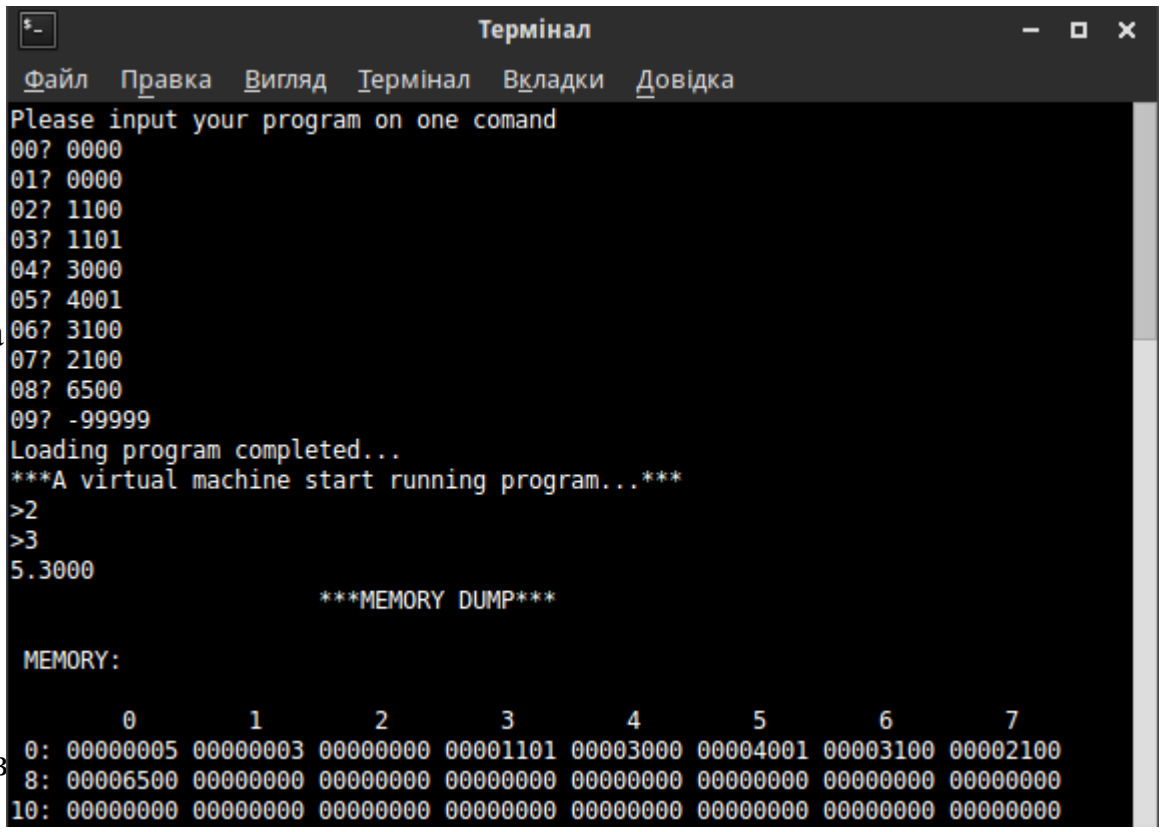
ZZ?

ZZ-це значення поточної адреси в інтервалі від 0 до ff (0..255).



Для того , щоб завершити введення команд , потрібно ввести контрольне значення -99999.
Увага! При завантаженні програми з файла контрольне значення вводити не потрібно.

Після введення команд, програма автоматично починає їх виконання. Приклад програми подано на наступному зображенні:



Якщо ви не хочете вводити команду за даною адресою, або хочете зарезервувати місце в пам’яті за даною адресою, запишіть за поточною адресою, 0000, або інше число представленому в 16-ричній системі числення (HEX), наприклад 000a або 000A , що є ідентичним.

Дамп пам’яті та регістрів

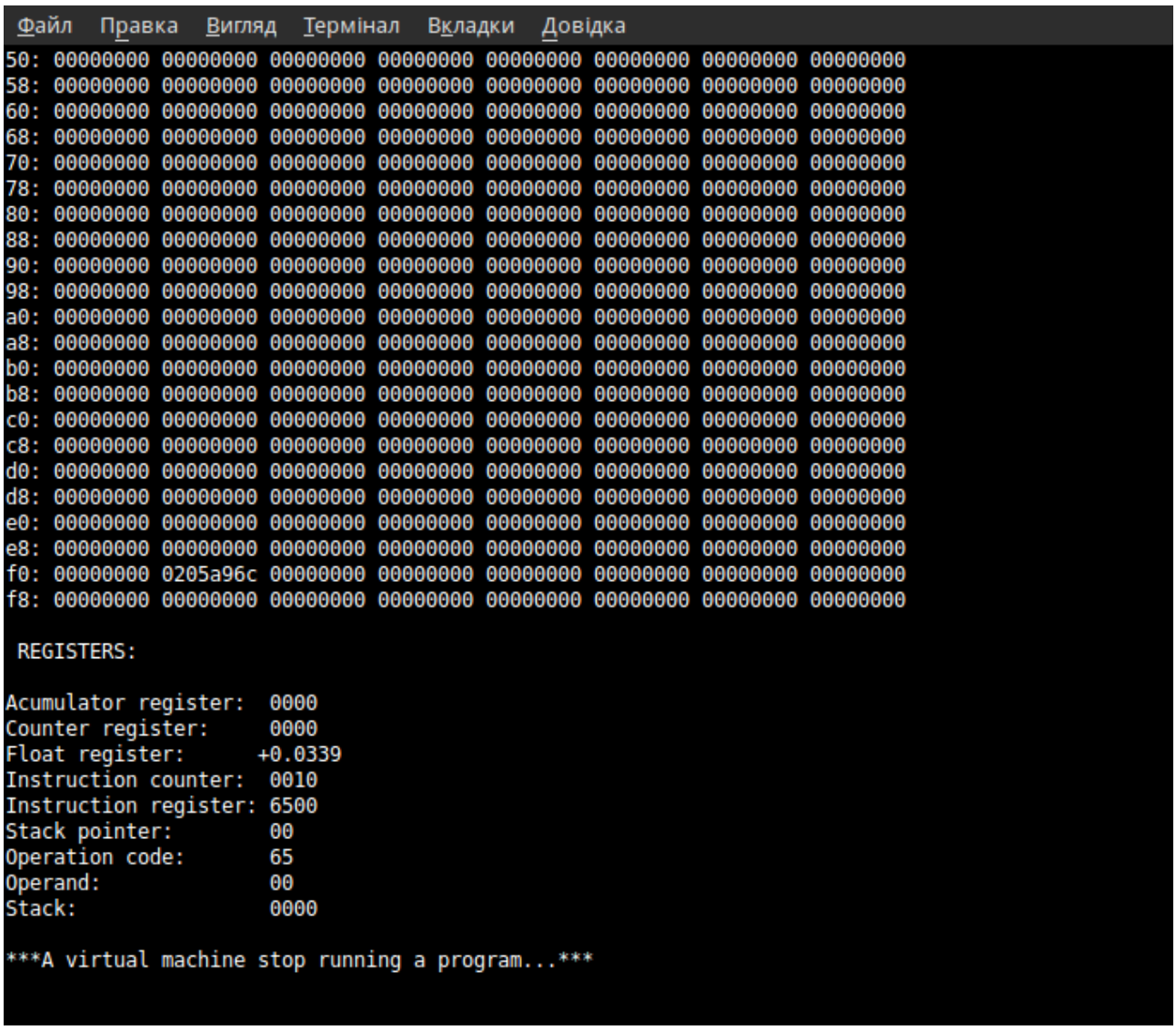
Якщо ви задасте виводити дамп пам’яті по завершенні роботи програми, це надасть змогу вам перевіряти які значення набували комірки пам’яті і регістрів на час завершення роботи програми. Варто зауважити, що дамп пам’яті автоматично виводиться у разі аварійного завершення роботи програми.
Формат дампу пам’яті і регістрів має такий вигляд:

Дамп оперативної пам’яті

```
Файл  Правка  Вигляд  Термінал  Вкладки  Довідка
                    ***MEMORY DUMP***

MEMORY:

       0        1        2        3        4        5        6        7
 0: 00000003 00000587 0000005c 005b8d71 00001100 00001102 00003200 00005202
 8: 000033f0 000021f0 00006500 00000000 00000000 00000000 00000000 00000000
10: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
18: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
20: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
28: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
30: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
38: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
40: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
48: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
50: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
58: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
60: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
68: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
70: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
78: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
80: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
88: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
90: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
98: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
a0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
a8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
b0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
b8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
c0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
c8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
d0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
d8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e0: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
e8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
f0: 00000000 0205a96c 00000000 00000000 00000000 00000000 00000000 00000000
f8: 00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```



Дамп значень регістрів і лічильників, та останнього значення стеку

Команди введення / виведення даних в (з пам'яті) пам'яті

У вихідному коді до програми симулятора в файлі (commands.h) надано перелік всіх констант та їх числових значень , що відповідають за певну команду. Вміст файлу такий:

```
/*Команди зчитування даних*/
#define READ_DEC     0x10
#define READ_FLOAT 0x11
#define READ_CHAR  0x12
#define READ_STR      0x13
#define READ_HEX     0x14

/*Команди виведення на консоль*/
#define WRITE_DEC     0x20
#define WRITE_FLOAT 0x21
#define WRITE_CHAR  0x22
#define WRITE_STR      0x23
#define WRITE_HEX     0x24

/*Команди завантаження/вивантаження даних*/
#define LOAD_ACUM   0x30
#define STORE_ACUM  0x31
#define LOAD_FREG     0x32
#define STORE_FREG    0x33
#define PUSH_STACK    0x34
#define POP_STACK      0x35
#define LOAD_CREG     0x36
#define STORE_CREG   0x37

/*Арифметичні операції з акумулятором*/
#define ACUM_ADD   0x40
#define ACUM_SUB   0x41
#define ACUM_DIV   0x42
#define ACUM_MOD   0x43
#define ACUM_MUL   0x44
#define ACUM_SHR   0x45
#define ACUM_SHL   0x46

/*Арифметичні операції з регістром обробки дійсних чисел*/
#define FREG_ADD  0x50
#define FREG_SUB  0x51
#define FREG_DIV  0x52
#define FREG_MUL  0x53
```

```
/*Команди передачі керування*/
#define BRANCH       0x60
#define BRANCH_ZERO  0x61
#define BRANCH_NEG   0x62
#define LOOP         0x63
#define HALT         0x64
#define HALT_INFO    0x65
```

Розглянемо детально кожну команду. Припустимо нам потрібно зчитати з консолі і записати ціле число за 11 адресою.
Для цього за поточною адресою вводимо 1011.
Вважатимемо, що XX- це наша адреса за якою ми хочемо записати,а ZZ? - поточна адреса.
Тоді наступні команди здійснюють такі дії:
ZZ? 10XX - читає з консолі і записує за адресою XX ціле число;

ZZ? 11XX - читає з консолі і записує починаючи з адреси XX дійсне число. (Зауваження. Дійсне число займає дві комірки пам'яті. В одній міститься ціла, а в іншій дробова частина.);

ZZ? 12XX - читає з консолі символ і заносить його ascii-код за адресою XX;

ZZ? 13XX — читає послідовність символів (рядок) і записує її починаючи з адреси XX;

ZZ? 14XX — зчитує hex-число з консолі і записує його за в пам'ять за адресою XX;

Наступна серія команд виводить на екран у відповідності до команди:
ZZ? 10XX — виводить ціле число ;
ZZ? 11XX- виводить дійсне число ;
ZZ? 12XX — виводить символ;
ZZ? 13XX — виводить рядок;
ZZ? 14XX — виводить hex-число;

Команди завантаження /вивантаження даних в регістри і стек (з регістрів і стеку)

Наступні команди призначені для завантаження даних з оперативної пам'яті для їх обробки і вивантаження їх з пам'яті після обробки.
Для завантаження(вивантаження) даних в(з) акумулятор(а) використовуються такі команди:
ZZ? 30XX — завантажити дані в акумулятор з області пам'яті за адресою XX;
ZZ? 31XX — вивантажити дані в пам'ять за адресою XX з акумулятора .

Для завантаження(вивантаження з регістру) даних в регістр дійсних чисел використовуються такі команди:
ZZ? 32XX — завантажити дані з пам'яті;
ZZ? 33XX — вивантажити дані з пам'яті.

Для завантаження даних в стек з області пам'яті за адресою XX використовується команда ZZ? 34XX.
Для вивантаження даних зі стеку в область пам'яті XX використовується команда ZZ? 35XX.

Для завантаження даних у регістр-лічильник з певної області пам'яті за адресою XX використовується команда ZZ? 36XX.
Для вивантаження даних в певну область пам'яті використовується команда ZZ? 37XX.

Арифметичні операції над акумулятором

Для обробки цілих чисел використовують акумулятор.
Для обробки даних в акумуляторі передбачені такі операції:
додавання, віднімання, множення , цілочисельне ділення, операція по модулю від цілочисельного ділення, побітовий зсув вправо, побітовий зсув вліво.
Відповідно коди арифметичних операцій такі:
ZZ? 40XX — додати значення з області пам'яті за адресою XX до значення в акумуляторі;
ZZ? 41XX- відняти аід значення в акумуляторі значення в пам'яті;
ZZ? 42XX- поділити значення в акумуляторі на значення в пам'яті;
ZZ? 43XX- знайти залишок від ділення значення в акумуляторі на значення в пам'яті;
ZZ? 44XX- домножити значення в акумуляторі на значення в пам'яті;
ZZ? 45XX- здійснити побітовий зсув вправо на значення що міститься в пам'яті;
ZZ? 46XX- здійснити побітовий зсув вліво на значення що міститься в пам'яті.

Арифметичні операції в над регістром дійсних чисел

Для обробки дійсних чисел використовуюють в даній моделі регістр дійсних чисел.
Для регістру дійсних чисел передбачені такі операції:
додавання;
віднімання;
множення;
ділення;
Принцип роботи регістру дійсних чисел схожий з роботою акумулятора.
Коди операцій для регістру такі:
ZZ? 50XX — додати значення з області пам’яті;
ZZ? 51XX - відняти значення з області пам’яті;
ZZ? 52XX — поділити на значення з області пам’яті;
ZZ? 53XX — домножити на значення з області пам’яті;

Команди передачі управління

Для передачі управління передбачені такі команди:
ZZ? 60XX — безумовний перехід до адреси XX;
ZZ? 61XX — перехід за адресою XX якщо в акумуляторі нуль;
ZZ? 62XX — умовний перехід за адресою XX якщо в акумуляторі від'ємне число.
ZZ? 63XX — перехід за адресою XX якщо в регістрі-лічильнику не нуль.
ZZ? 6400 — вихід з програми.
ZZ? 6500 — вихід з програми з виводом дампу пам'яті.