Віртуальна машина СM-1

(Довідник користувача)

Віртуальна машина CM-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 .

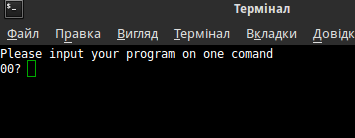
YY -числове значення в 16-ричній системі числення , що відповідає адресі , за якою потрібно здійснити дію.

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).

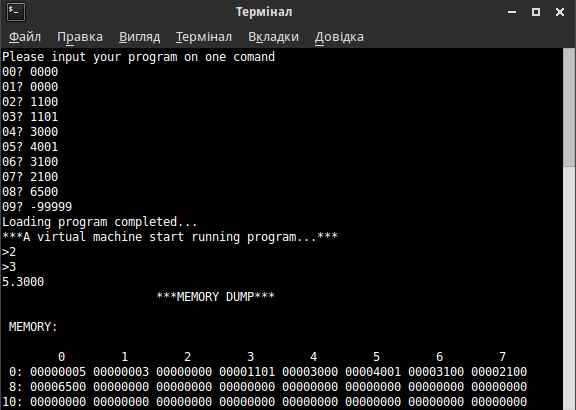


Для того , щоб завершити введення команд , потрібно ввести контрольне значення -99999.

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:

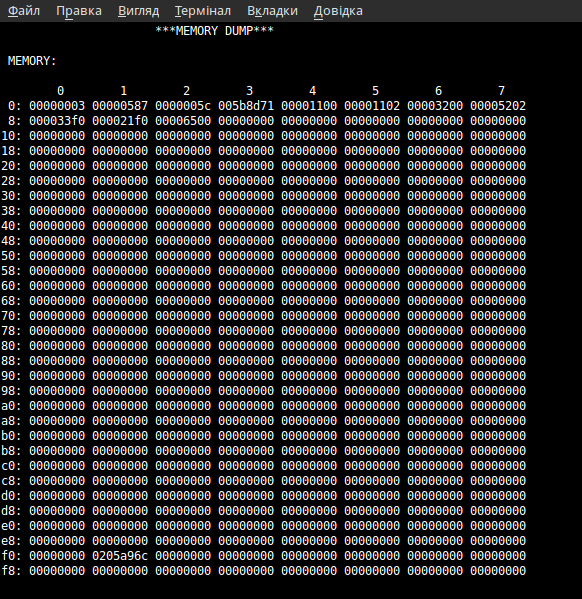


Якщо ви не хочете вводити команду за даною адресою, або хочете зарезервувати місце в пам’яті за даною адресою, запишіть за поточною адресою, 0000, або інше число представленому в 16-ричній системі числення (HEX), наприклад 000a або 000А , що є ідентичним.

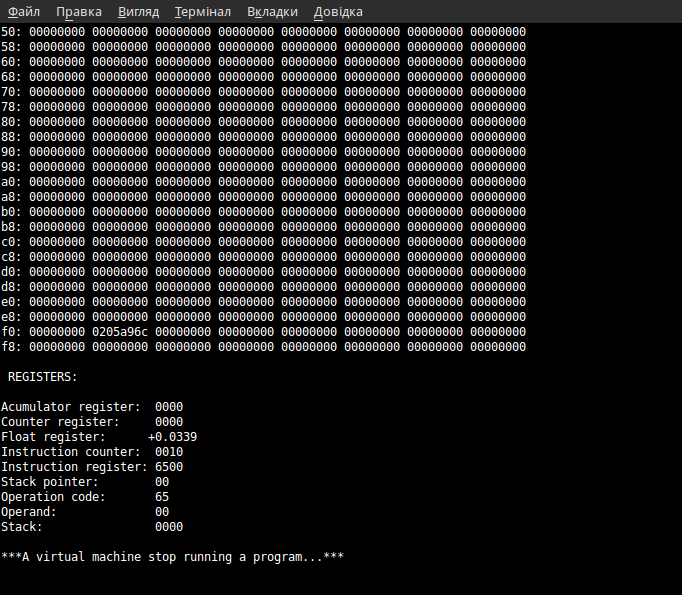
**Дамп пам’яті та регістрів**

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*



*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,a 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? 32ХХ — завантажити дані з пам’яті;

ZZ? 33ХХ — вивантажити дані з пам’яті.

Для завантаження даних в стек з області пам’яті за адресою ХХ використовується команда ZZ? 34ХХ.

Для вивантаження даних зі стеку в область пам’яті ХХ використовується команда ZZ? 35ХХ.

Для завантаження даних у регістр-лічильник з певної області пам’яті за адресою ХХ використовується команда ZZ? 36ХХ.

Для вивантаження даних в певну область пам’яті використовується команда

ZZ? 37ХХ.

**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? 40ХХ — додати значення з області пам’яті за адресою ХХ до значення в акумуляторі;

ZZ? 41ХХ- відняти аід значення в акумуляторі значення в пам’яті;

ZZ? 42ХХ- поділити значення в акумуляторі на значення в пам’яті;

ZZ? 43ХХ- знайти залишок від ділення значення в акумуляторі на значення в пам’яті;

ZZ? 44ХХ- домножити значення в акумуляторі на значення в пам’яті;

ZZ? 45ХХ- здійснити побітовий зсув вправо на значення що міститься в пам’яті;

ZZ? 46ХХ- 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? 50ХХ — додати значення з області пам’яті;

ZZ? 51ХХ - відняти значення з області пам’яті;

ZZ? 52ХХ — поділити на значення з області пам’яті;

ZZ? 53ХХ — домножити на значення з області пам’яті;

**Команди передачі управління**

Для передачі управління передбачені такі команди:

ZZ? 60ХХ — безумовний перехід до адреси XX;

ZZ? 61XX — перехід за адресою ХХ якщо в акумуляторі нуль;

ZZ? 62XX — умовний перехід за адресою ХХ якщо в акумуляторі від’ємне число.

ZZ? 63XX — перехід за адресою ХХ якщо в регістрі-лічильнику не нуль.

ZZ? 6400 — вихід з програми.

ZZ? 6500 — вихід з програми з виводом дампу пам’яті.