**과목명: 시스템프로그래밍**

**02 분반**

**<<Project #1>>**

**서강대학교 컴퓨터공학과**

**20181614**

**김주연**

목 차

1. **프로그램 개요**
   1. 프로젝트 문제
   2. 프로젝트 목표
2. **프로그램 설명**
   1. 프로그램 흐름도
   2. 프로그램 설명
3. **모듈 정의**
   1. 정의한 모듈에 대해 간략한 설명
4. **전역 변수 정의**
5. **코드 설명**
6. **프로그램 개요**

1.1 프로젝트 문제

이 프로그램은 앞으로 구현하게 될 SIC/XE 머신을 구현하기 위한 전 단계로서, 어셈블러, 링커, 로더들을 실행하게 될 셸(Shell)과 컴파일을 통해서 만들어진 object 코드가 적재되고 실행될 메모리 공간과 mnemonic(ADD, COMP, FLOAT, etc.)을 opcode 값으로 변환하는 OPCODE 테이블과 관련 명령어들을 구현하는 프로그램이다.

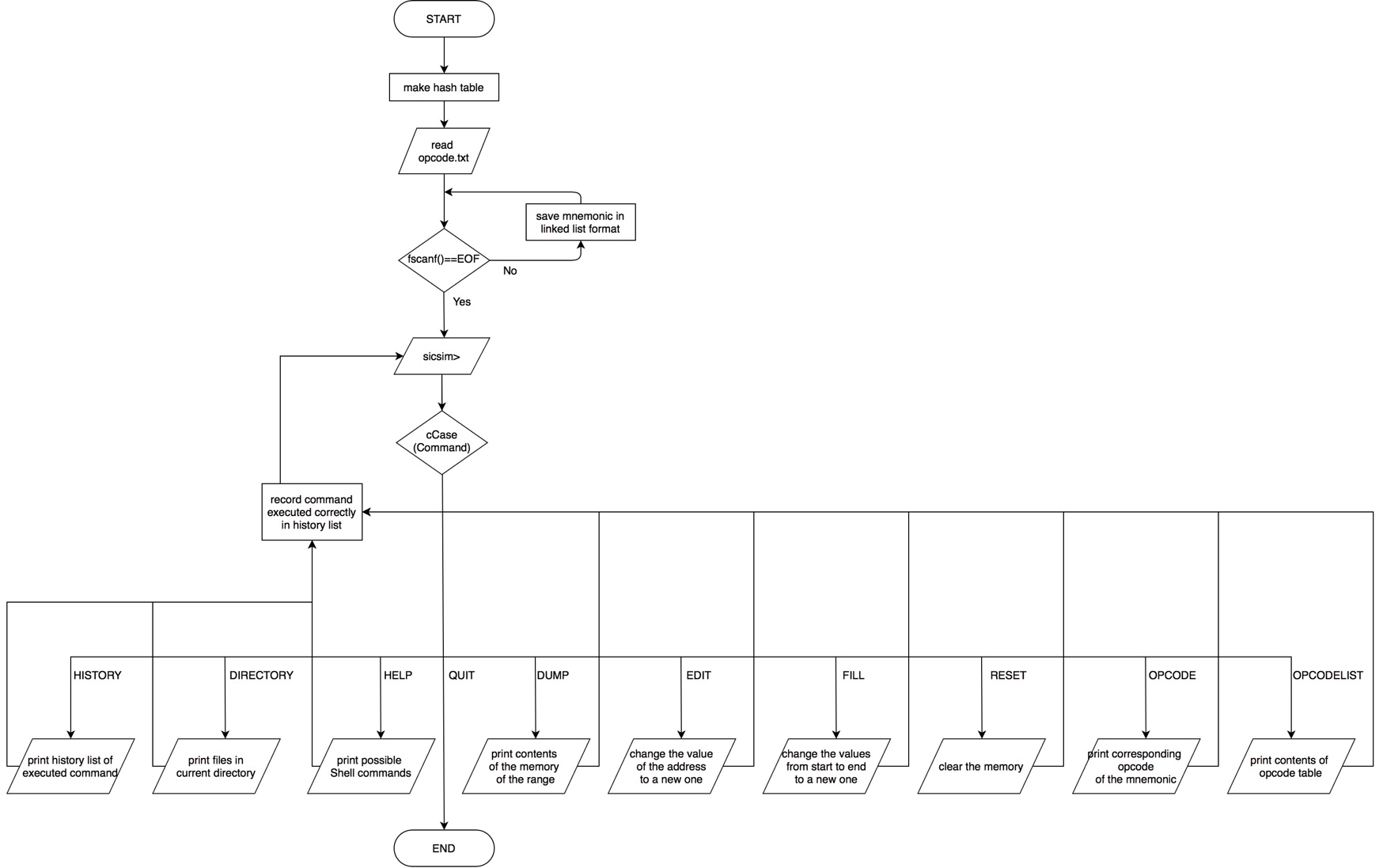
1.2 프로젝트 목표

이 프로그램을 실행시키면 아래와 같이 unix shell과 유사한 입력 프롬프트 상태가 된다.

sicsim>

이 상태에서 아래에 있는 명령어들을 입력할 때 그에 해당되는 기능을 수행한다.

* 셸 관련 명령어들 (help, dir, quit, history)
* 메모리 공간 관련 명령어들 (dump, edit, fill, reset)
* opcode 관련 명령어들 (opcode, opcodelist)

1. **프로그램 설명**
   1. 프로그램 흐름도

이는 프로그램의 전체적인 흐름도이다.

* 1. 프로그램 설명

프로그램이 실행되면, size가 20인 hash table 포인터 구조체를 선언하여 초기화한다. 이후 opcode.txt에서 mnemonic의 list를 받아 와 그에 맞는 opcode와 format을 구조체에 기록하여 hash table에 연결한다(makeHash()에서 hash table 생성, 초기화한 후 opcode.txt에서 mnemonic list를 한 줄 씩 읽어 와 hashConnect()에서 이를 linked list 형태로 연결).

이후 이 프로그램에서 사용할 1MB size의 가상 메모리 dumpMem[][]과 history 기록 저장용 변수 등의 전역변수 초기화 작업을 수행한다.

그 이후에는 sicsim 프롬프트에서 입력을 받아 입력으로 들어 온 command에 따른 기능(help, dir, quit, history, dump, edit, fill, reset, opcode, opcodelist)을 수행하게 되며, quit(or q)이 입력으로 들어오기 전까지 반복해서 command line을 입력받는다. 입력받은 command line은 comma 제거, command만 분리 등 작업을 거치게 되며, 그 이후 command는 switch문을 통해 구분되어 그에 맞는 함수를 호출하게 된다. 각 함수에서는 정해진 기능(help, dir, history, dump, edit, fill, reset, opcode, opcodelist)을 수행한다. 이 때 Enum을 이용하여 선택지를 간단하게 나타냄으로서 가독성을 높였다.

또한, 프로그램은 4개의 c 파일에 나누어 작성하였다.

20181614.c는 메인 파일의 역할을 하며, hash table 생성함수 호출, 전역변수 초기화, 쉘 명령어 입력과 그에 해당하는 함수 호출 등의 기능을 담고 있다. shell.c는 쉘 관련 명령어들인 help, dir, history의 기능을 수행하는 함수와 history list를 기록하는 linked list를 작성하는 함수를 담고 있다. memory.c는 메모리 공간 관련 명령어들인 dump, edit, fill, reset 기능을 수행하는 함수를 담고 있다. opcode.c는 opcode관련 명령어들인 opcode, opcodelist 기능을 수행하는 함수를 담고 있다.

1. **모듈 정의**
   1. 정의한 모듈에 대해 간략한 설명

20181614.c 파일

* main() :

parameter : void

return : 프로그램 종료 시 0을 return한다.

기능 : makeHash() 함수를 호출하여 mnemonic을 저장할 hash table을 초기화/생성한다. history 출력을 위해 command history를 기록할 linked list의 개수와 시작 지점을 초기화한다. 메모리 관련 함수에서 사용할 1MB size의 가상 메모리를 초기화하고, dump 기능에서 마지막으로 출력한 주소를 기록하는 변수 addrSave를 초기화한다. while문을 이용하여 sicsim 프롬프트에서 계속해서 command를 입력받는다. regular\_form() 함수를 호출하여 입력받은 command line에서 comma를 제거하고, strtok를 사용하여 line에서 command만 분리한 뒤 commandCase() 함수를 통해 해당 command에 해당하는 번호를 반환받는다. 번호는 20181614.h 헤더파일에서 Enum으로 지정해놓은 키워드를 통해 switch문에 사용된다. 각 case마다 해당하는 기능을 하는 함수를 호출하게 되며, 기능이 정상적으로 수행된 경우 hisConnect() 함수를 호출하여 입력되었던 command line을 history list에 기록한다.

함수 호출 시, dump, edit, fill, opcode 등의 명령어는 추가로 메모리 주소나 value값, mnemonic 값을 입력받으므로 strtok를 적절하게 사용하여 함수가 제대로 수행될 수 있도록 인자로 주었다.

* regular\_form() :

parameter : 사용자가 입력한 command string

return : command string에서 ,(comma)를 제거한 새로운 string을 return한다.

기능 : 사용자가 입력한 command string 크기의 새로운 string을 생성하고, for문을 통해 command에서 새로운 string으로 문장을 옮긴다. 이때 만약 원 command에 comma가 있다면 그것을 제거하고 옮긴다. 문장을 다 옮긴 후에는 그것을 저장할 새로운 string을 생성하여 string 끝에 ‘\0’을 삽입하고 해당 string을 return한다.

* commandCase() :

parameter : command line에서 strtok를 사용하여 command만 분리한 string

return : function에 맞는 숫자를 return한다.

기능 : main()에서 strtok를 사용해 사용자가 입력한 command line에서 command만 분리한 string을 입력으로 받아 strcmp를 사용하여 command의 종류에 따라 다른 숫자를 return한다. 만약 해당 command가 존재하지 않는 command라면 -1을 return하여 main에서 처리하지 않도록 한다.

shell.c 파일

* hFunc() :

parameter : void

return : void

기능 : 이 프로그램에서 수행할 수 있는 기능들의 리스트를 출력한다.

* dFunc() :

parameter : void

return : 수행이 제대로 되었다면 0을, 오류가 생긴다면 -1을 return한다.

기능: 현재 디렉토리에 있는 파일들을 받아서 파일들의 상태를 확인하고, 만약 해당 파일이 디렉토리 파일이라면 파일명 뒤에 ‘/’을 더해서 출력, 해당 파일이 실행 파일이라면 파일명 뒤에 ‘\*’을 더해서 출력한다. 실행 파일을 판단하기 위해 파일 이름을 strtok를 사용하여 .을 기준으로 잘라 .뒤에 오는 확장자가 out인지를 확인하는 방식을 사용하였다.

* hisConnect() :

parameter : 사용자가 입력한 command line string

return : void

기능: 새로운 history를 기록할 struct를 초기화하고, history의 개수를 저장하는 전역 변수 hisCnt와 사용자가 입력한 command line을 내용으로 하여 history struct를 채운다. 그 후 for문을 사용하여 현재 저장하고 있는 history linked list의 끝에 새로운 history struct를 추가한다.

* hiFunc() :

parameter : void

return : void

기능: history를 기록하고 있는 linked list를 이용하여 현재까지 사용자가 입력한 모든 command들을 순서대로 보여준다. for문을 사용하여 linked list의 내용을 순서대로 출력하였다.

memory.c 파일

* duFunc() :

parameter : 사용자가 입력한 start, end string 값

return : 수행이 제대로 되었다면 1을, 오류가 생긴다면 0을 return 한다.

기능: dump기능은 3가지 종류가 있는데, 이는 dump, dump start, dump start, end의 3가지이다. 3가지 기능을 duFunc()함수에서 모두 수행하도록 하였다.

우선 start, end string을 받아왔을 때, start와 end가 모두 null이라면 사용자가 dump를 입력한 경우임을 알 수 있다. 이 때의 start 주소는 이전에 dump를 수행했던 마지막 주소(addrSave)+1값이며, end값은 start를 포함하여 160개를 출력하는 주소이어야 하므로 start+0x9F가 된다. 이때, 만약 end 값이 메모리의 한계값인 0xFFFFF를 초과한다면 0xFFFFF까지만 출력하고, dump 주소를 저장하는 addrSave 변수를 0x00으로 초기화한다.

만약 start는 있는데 end가 null이라면 사용자가 dump start를 입력한 경우임을 알 수 있다. 이 때의 start주소는 입력받은 start string을 strtol을 사용하여 16진수로 변환한 값이며, start 주소가 0xFFFFF를 초과하거나 negative값을 갖는 경우에는 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였다. 이때의 end값은 start를 포함하여 160개를 출력하는 주소이므로 start값에 0x9F를 더한 값을 end값으로 하였다. end값이 만약 한계값인 0xFFFFF를 초과한다면 0xFFFFF까지만 출력되도록 하였다.

그리고 start와 end값이 모두 있는 경우라면 사용자가 dump start, end를 입력한 경우임을 알 수 있다. 이 때의 start와 end 주소는 입력받은 start, end string을 strtol을 사용하여 16진수로 변환한 값이며, start주소가 end 주소보다 큰 경우 오류로 처리하여 0을 반환하고 함수를 종료하였다. 또한 위에서와 마찬가지로 start 주소가 0xFFFFF를 초과하거나 negative값을 갖는 경우에는 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였으며, end 주소 또한 0xFFFFF를 초과하면 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였다(negative값을 갖는 경우는 start>end인 경우로 처리할 수 있다.)

이처럼 start와 end 주소를 결정한 뒤 해당 범위의 dumpMem 내용을 출력하는 기능을 수행한다. dumpMem 배열은 한 row에 16개의 column을 가지므로, 입력받은 start를 16으로 나눈 몫이 row가 된다. 이후 for문을 통해 start부터 end까지 주소값을 증가시키며 해당하는 메모리의 값을 출력하는 기능을 수행하였으며, 해당 row에 속하지만 start-end 범위에 해당하지 않는 메모리 주소의 값은 16진수 대신 빈칸으로 출력하였다. 또한 하나의 row에 해당하는 16개의 문자들을 저장할 배열 cArr를 선언하여 메모리 값을 저장하고, 16진수 출력값의 우측에 배열의 ASCII code 값을 character로 출력하였다. 20~7E 범위 내의 값만 출력되도록 하였으며, 이외의 값은 ‘.’으로 출력하였다.

* eFunc() :

parameter : 사용자가 입력한 address, value string 값

return : 수행이 제대로 되었다면 1을, 오류가 생긴다면 0을 return 한다.

기능: 입력받은 address, value string을 strtol을 사용하여 16진수로 변환하여 주소값과 value값을 구한다. 이 때, address 주소가 0xFFFFF를 초과하거나 negative값을 갖는 경우에는 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였으며, value 값이 1byte범위를 초과한 경우, 즉 0xFF를 초과한 경우나 negative인 경우 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였다.

이처럼 address와 value값을 결정한 뒤, 해당 address에 해당하는 메모리 값을 value로 바꿔 주는 작업을 수행한다. dumpMem 배열은 한 row에 16개의 column을 가지므로, 입력받은 address를 16으로 나눈 몫이 row가 되며, address를 16으로 나눈 나머지가 col이 된다. 구해진 row, col에 해당하는 메모리 주소의 값을 value로 수정하였다.

* fFunc() :

parameter : 사용자가 입력한 address, value string 값

return : 수행이 제대로 되었다면 1을, 오류가 생긴다면 0을 return 한다.

기능: 입력받은 start, end, value string을 strtol을 사용하여 16진수로 변환하여 주소값과 value값을 구한다. 이 때, start주소가 end 주소보다 큰 경우 오류로 처리하여 0을 반환하고 함수를 종료하였다. 또한 위에서와 마찬가지로 start 주소가 0xFFFFF를 초과하거나 negative값을 갖는 경우에는 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였으며, end 주소 또한 0xFFFFF를 초과하면 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였다(negative값을 갖는 경우는 start>end인 경우로 처리할 수 있다.) 또한 value 값이 1byte범위를 초과한 경우, 즉 0xFF를 초과한 경우나 negative인 경우 오류로 처리하여 0을 반환하고 함수를 종료하도록 하였다.

이처럼 start와 end, value값을 결정한 뒤, start번지 부터 end번지에 해당하는 메모리 값을 모두 value로 바꿔 주는 작업을 수행한다. dumpMem 배열은 한 row에 16개의 column을 가지므로 입력받은 address를 16으로 나눈 몫이 row가 되며, 이를 번지값으로 바꾼 row\*16+column(i 값)이 범위에 해당하면 메모리 주소의 값을 value로 수정하였다.

* resetFunc() :

parameter : void

return : void

기능 : 프로그램에서 선언한 1MB 가상 메모리 dumpMem[][]에 저장된 모든 값을 0x00으로 초기화한다.

opcode.c 파일

* hashConnect() :

parameter : opcode.txt에서 받아 온 opcode, mnemonic, format 값

return : void

기능 : 사용자가 opcode.txt에서 opcode, mnemonic, format을 한 줄 씩 읽어 와 받은 값을 opcode를 저장하는 hash table에 linked list 형태로 저장한다. hash table struct를 초기화하고 받아온 값을 그 struct에 넣어 준다. 이때, 3/4형식이 가능한 mnemonic의 format은 3으로 통일해 주었다. struct를 완성한 뒤, makeHash() 함수에서 선언해 놓은 hash table op에 연결해 주었는데, mnemonic의 첫 알파벳을 20으로 나눈 값을 해당 mnemonic의 index로 하여 같은 알파벳이 같은 index에 담겨있도록 함으로써 추후 mnemonic 검색의 효율을 높였다.

* makeHash() :
* parameter : void
* return : 수행이 제대로 되었다면 0을, 오류 발생 시 -1을 return한다.
* 기능 : opcode를 저장할 size 20의 hash table을 선언하고 초기화한다. 그리고 opcode.txt 파일을 열어 저장된 mnemonic list를 한 줄 씩 읽어 와 그것들을 인자로 하는 hashConnect() 함수를 호출한다. txt파일을 다 읽은 뒤 파일을 닫는다.
* opMnemFunc() :

parameter : 사용자가 입력한 mnemonic

return : 수행이 제대로 되었다면 1을, 해당 mnemonic이 없다면 0을 return한다.

기능 : 사용자가 입력한 mnemonic을 입력으로 받아 해당 mnemonic의 첫 알파벳을 20으로 나눈 값을 찾는다. 해당 값이 opcode hash table의 index 주소가 되므로, hash table pointer를 통해 해당 index의 처음부터 linked list를 훑으며 mnemonic의 opcode를 검색하고, opcode가 존재한다면 mnemonic과 함께 출력한다. 만약 사용자가 입력한 mnemonic이 존재하지 않는다면 0을 리턴한다.

* opListFunc() :

parameter : void

return : void

기능 : 프로그램 시작 시 저장했던 opcode hash table의 내용을 출력한다. for문을 통해 index 0부터 19까지 출력하게 되며, hash table pointer를 통해 linked list를 차례대로 출력하게 된다. 만약 해당 index에 어떤 mnemonic도 존재하지 않는다면 아무 것도 출력하지 않는다.

1. **전역 변수 정의**

* enum commEnum

: 사용자가 입력한 command에 따라서 commandCase() 함수가 해당하는 기능에 따라 다른 숫자를 return하게 되는데, return하는 숫자의 순서에 맞추어 enum에서 해당 command를 매칭해두어 main()에서의 switch기능 수행 시 보다 readable하도록 하였다(ex> case 1 대신 case HELP의 형태로 사용 가능).

* struct historyList

: 사용자가 입력한 command line을 linked list 형태로 기록하기 위한 구조체이다. 입력한 순서를 기록하는 no, command line을 기록하는 command[30], 다음 구조체 포인터 link를 가진다.

* HLIST\* hisHead

: 사용자의 입력 history를 기록하는 linked list의 첫 부분이다.

* int hisCnt

: 사용자의 입력 history 기록을 위해 현재 command의 입력 순서를 기록하는 변수이다.

* int addrSave

: dump 기능 시 전에 dump를 수행했던 주소의 바로 다음 주소부터 프린트를 하게 되는데, 이때 전에 dump를 수행했던 주소를 기록하는 변수이다.

* unsigned char dumpMem[65536][16]

: 이 프로그램에서 사용할 1MB 가상 메모리를 unsigned char 타입의 이차원 배열을 선언하여 지정하였다.

* struct hashTable

: 프로그램을 시작할 때 opcode.txt에서 mnemonic과 opcode, format을 linked list 형태의 hash table로 구현하기 위한 구조체이다. opcode를 기록할 opcode, mnemonic을 기록할 mnemonic[10], format을 기록할 format, 그리고 다음 구조체 포인터 link를 가진다.

* HTABLE\*\* op

: hash table 구현을 위해 사용할 linked list의 첫 부분이다.

1. **코드 설명**

코드 설명은 위의 모듈 설명에 더해 주석으로 하였다.

* 20181614.c 파일

텍스트, 스크린샷, 노트북, 앉아있는이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명 전화, 휴대폰, 노트북, 모니터이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

* shell.c 파일

스크린샷, 텍스트이(가) 표시된 사진

자동 생성된 설명스크린샷이(가) 표시된 사진

자동 생성된 설명

* memory.c 파일

텍스트, 앉아있는, 전화, 휴대폰이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명



텍스트, 전화, 앉아있는, 휴대폰이(가) 표시된 사진

자동 생성된 설명

텍스트, 전화이(가) 표시된 사진

자동 생성된 설명

그리기이(가) 표시된 사진

자동 생성된 설명

* opcode.c 파일

스크린샷, 전화이(가) 표시된 사진

자동 생성된 설명

스크린샷이(가) 표시된 사진

자동 생성된 설명

* 20181614.h 파일

스크린샷이(가) 표시된 사진

자동 생성된 설명

