ARMsim

CSC 236

ARMsim

- ARMSim
  - Assembler
  - Simulator
  - Download and install
- SWI
  - Software interrupt
  - Same idea as 8086 int 21h
- File I/O
  - We won't do terminal input (but we could if we wanted to)

SWI

SWI		Input	Output
0x66	Open file	r0 = ptr file name r1 = 0 (in) or 1 (out)	r0 = file handle
0x68	Close file	r0 = file handle	
0x69	Write string	r0 = file handle r1 = ptr to string	
0x6a	Read string	r0 = file handle r1 = ptr to string r2 = max bytes	r0 = # written

SWI

SWI		Input	Output
0x66	Open file	r0 = ptr file name r1 = 0 (in) or 1 (out)	r0 = file handle
0x68	Close file	r0 = file handle	
0x69	Write string	r0 = file handle r1 = ptr to string	
0x6a	Read string	r0 = file handle r1 = ptr to string r2 = max bytes	r0 = # written

- In — read
  - File must exist (and be readable)
- Out — write
  - Creates file if it doesn't exist
- Returns file handle

SWI

SWI		Input	Output
0x66	Open file	r0 = ptr file name r1 = 0 (in) or 1 (out)	r0 = file handle
0x68	Close file	r0 = file handle	
0x69	Write string	r0 = file handle r1 = ptr to string	
0x6a	Read string	r0 = file handle r1 = ptr to string r2 = max bytes	r0 = # written

- Closes files
  - File is no longer usable
  - Flushing data in buffer (if write)

SWI

SWI		Input	Output
0x66	Open file	r0 = ptr file name r1 = 0 (in) or 1 (out)	r0 = file handle
0x68	Close file	r0 = file handle	
0x69	Write string	r0 = file handle r1 = ptr to string	
0x6a	Read string	r0 = file handle r1 = ptr to string r2 = max bytes	r0 = # written

- Write string
  - EOS = 00h

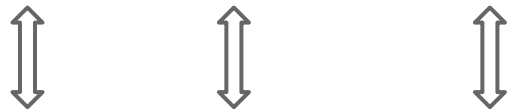
SWI

SWI		Input	Output
0x66	Open file	r0 = ptr file name r1 = 0 (in) or 1 (out)	r0 = file handle
0x68	Close file	r0 = file handle	
0x69	Write string	r0 = file handle r1 = ptr to string	
0x6a	Read string	r0 = file handle r1 = ptr to string r2 = max bytes	r0 = bytes read

- Read string
  - Stops at max bytes
  - If return count = 0 — at end of file
- EOL (either \r\n or \n)
  - Replaced by 00h

Data declarations

```
v1:   .skip       4           ;reserve 4 bytes
```



label **format** **value**

Data declarations

```
v1:   .skip       4              ;reserve 4 bytes
v2:   .word       1000           ;32 bit word
v3:   .word       0x000003e8     ;hex
```

Data declarations

```
v1:   .skip       4           ;reserve 4 bytes
v2:   .word       1000        ;32 bit word
v3:   .word       0x000003e8     ;hex
v4:   .hword      555         ;half word
```

Data declarations

```
v1:   .skip       4           ;reserve 4 bytes
v2:   .word       1000        ;32 bit word
v3:   .word       0x000003e8     ;hex
v4:   .hword      555         ;half word
v5:   .byte       10          ;byte
v6:   .byte       -10         ;another byte
```

Data declarations

```
v1:   .skip       4           ;reserve 4 bytes
v2:   .word       1000        ;32 bit word
v3:   .word       0x000003e8     ;hex
v4:   .hword      555         ;half word
v5:   .byte       10          ;byte
v6:   .byte       -10         ;another byte
v7:   .asciz      "ABC"       ;ASCII string
                  ;terminated with 00h
v8:   .ascii      "DEF"       ;ASCII string
                  ;not terminated with 00h
```

Sample programs

- Three provided
- Demonstrate all needed directives
  - Define data
- And system calls
  - Open and close a file
  - Read string
  - Write a string
  - Terminate

- hello.s
  - Write ‘hello world’
  - To ‘HELLO.OUT’
- copystr.s
  - Reads string from DATA.IN
  - Copies string to new_string
  - Writes new_string to DATA.OUT
- copyfile.s
  - Copies contents of FILE.IN
  - To FILE.OUT