ECOAR Lab – general info

Grzegorz Mazur

Institute of Computer Science
Warsaw University of Technology

October 2018

Simple MIPS assembly project – exercise 1

Write a simple MIPS assembly program running in MARS simulator environment, processing a string read from console and producing desired output. The project must be completed in 2 hours during the lab session. The task will be similar to the ones listed below. Input string should be read from the console using read_string function.

- Write a program replacing all digits in a string with their complement to 9 ($0 \rightarrow 9$, $1 \rightarrow 8$, $2 \rightarrow 7$ etc.)
- Write a program displaying the longest sequence of digits found in a string.
- Reverse the order of digits in a string.
- Scan the biggest unsigned decimal number found in a string and display its value using print_int function.
- Display the number of decimal numbers (sequences of decimal digits) found in a string.
- Remove all digits from a string, display the resulting string.
- Remove from a string the sequence of characters specified by positions entered as integers; handle correctly out-of-range cases and both possible orders of positions:
    - “abcdefgh” 2 4 or “abcdefgh” 4 2 should produce “abfgh”
    - “abcdefgh” 90 100 should produce “abcdefgh”
    - “abcdefgh” 10 5 should produce “abcde”
- Remove characters preceded by digits: “abc5f67gh” → “abc56h”
- change every 3rd lowercase letter to uppercase: “ab1cde2f3gh4ij” → “ab1Cde2Fgh4Ij”
- remove all letters but the first from each sequence of uppercase letters: “ABCdefGHi” → AdefGi”

Simple x86 hybrid project – exercise 2

Write a simple x86 hybrid program with main module written in C and a function written in x86 assembly language (NASM assembler). The assembly module must conform to C calling convention as described in x86 System V ABI.

The routine should process the text string passed to it. The project must be completed in 2 hours during the lab session.

The task may be similar to the ones listed below.

```
char *removerng(char *s, char a, char b);
```
Remove characters with codes from a to b, a < b.

```
char *remnth(char *s, int n);
```
Remove every n/th character.

```
char *leavelastndig(char *s, int n);
```
Leave last n digits, rmoving all other characters

```
char *remrep(char *s);
```
Remove repetitions of characters.

```
char *leavelongestnum(char *s, int n);
```
Leave only the longest sequence of decimal digits.

```
char *leaverng(char *s, char a, char b);
```
Leave characters with codes from a to b, a < b.

```
char *remlastnum(char *s);
```
Remove the last sequence of decimal digits.

```
unsigned int getdec(char *s);
```
Scan the first unsigned decimal number.

```
unsigned int gethex(char *s);
```
Scan the first hexadecimal number.

```
char *reversedig(char *s);
```
Reverse the order of digits, leaving the other characters in their places.

```
char *reverselet(char *s);
```
Reverse the order of leters, leaving the other characters in their places.

```
char *reversepairs(char *s);
```
Swap characters in pairs

```
char *replnum(char *s, char a);
```
Replace each sequence of digits with a specified single character.

```
char *capwords(char *s);
```
Capitalize the first character of each word in a string.

MIPS project – exercise 3

Write a program in MIPS assembly using MARS or other SPIM-compatible simulator. The program should not rely on uninitialized register values and should be able to be run more than once without reloading. Avoid the sequences of consecutive branches, esp. conditional branches followed by unconditional ones (unless necessary).
All the text processing programs with file i/o should define getc and putc functions providing proper buffering of input and input operations with at least 512-byte buffers.
Graphic programs should display images using MARS graphic display mapped to heap address range.
Maximum score is 6 points. The project may be shown during two lab sessions. Each week of delay started will introduce a penalty of one point.
Projects are published in a separate file.

x86 projects

Write a program containing two source files: main program written in C and assembly module callable from C. The C declaration of an assembly routine is given for each project task. Use NASM assembler (nasm.sf.net) to assemble the assembly module. Use C compiler driver to compile C module and link it with the output of assembler. The C program should use command line arguments to supply the parameters to an assembly routine and perform all I/O operations. No system functions nor C library functions should be called from assembly code. Arguments for bit manipulation routines should be entered in hexadecimal.
Routines processing .BMP files may receive as arguments either the pointer to the whole .BMP file image in memory or the pointer to bitmap and its sizes read by main program. The routines should correctly process images of any sizes unless stated otherwise.
The program should be implemented in two versions: 32-bit, following the calling convention described in abi386-4.pdf and 64-bit, conforming to 64-bit Unix calling convention (document available from www.amd64.org). Both conventions are also described in a document available from www.agner.org. While converting from x86 to x86-64, try to remove memory variables if they were used in x86 version and place the variables in extra processor's registers.
Maximum score for (any) single version is 6 points. The second version is worth 2 points.

Avoid the sequences of consecutive branches, esp. conditional branches followed by unconditional ones (unless necessary).
Any attempt to submit the project explicitly violating the calling convention will be punished by subtracting one point from the final score.