Отчёта по лабораторной работе 6

Архитектура компютера

Волчкова Елизавета Дмитриевна

Содержание

Цель работы

Узнать и ознакомиться с арифметическими инструкциямиязыкаассемблера NASM.

Задание

1. Написать программу вычисления выражения y = f(x)

1.1. Программа должна выводить выражение для вычисления, выводитьзапрос на ввод значения x, вычислять заданное выражениевзависимости от введенного x, выводить результат вычислений.

1.2. Вид функции f(x) выбрать из таблицы 6.3 вариантовзаданийвсоответствии с номером полученным при выполнениилабораторнойработы.

1.3. Создайте исполняемый файл и проверьте его работу для значений x1и x2 из 6.3 # Теоретическое введение

Адресация в NASM Большинство инструкций на языкеассемблератребуют обработки операндов. Адрес операнда предоставляетместо, гдехранятся данные, подлежащие обработке. Это могут бытьданныехранящиеся в регистре или в ячейке памяти. Далее рассмотрены все существующие способызаданияадресахранения операндов – способы адресации. Существует три основных способа адресации:

• Регистровая адресация – операнды хранятся в регистрахивкомандеиспользуются имена этих регистров, например: mov ax,bx.

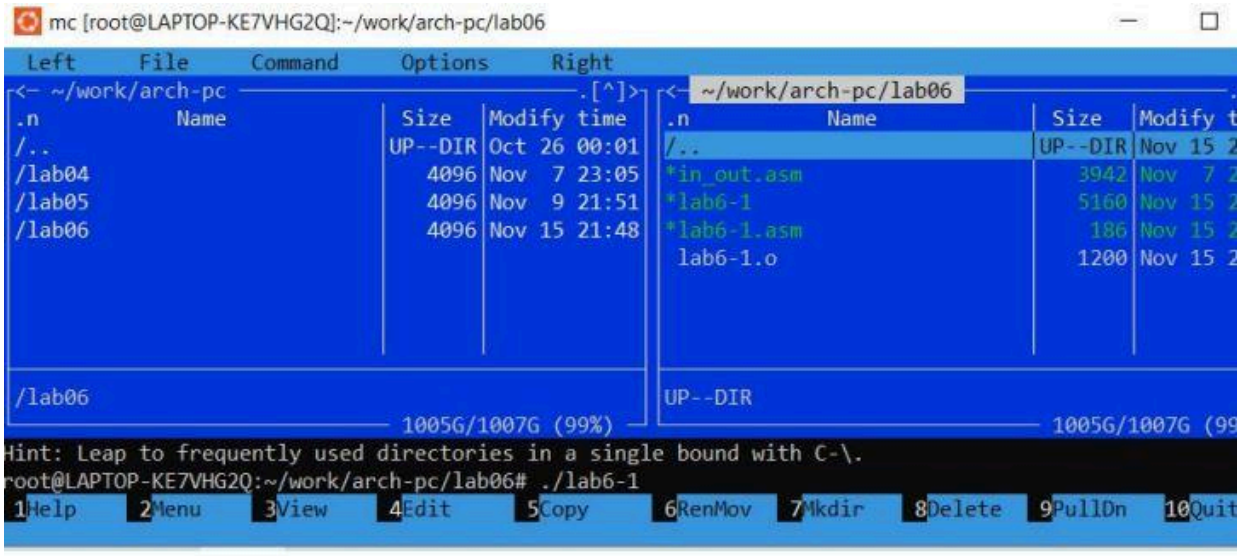
• Непосредственная адресация – значение операндазадаетсянепосредственно в команде, Например: mov ax,2.

• Адресация памяти – операнд задает адрес в памяти. Вкомандеуказывается символическое обозначение ячейки памяти, надсодержимымкоторой требуется выполнить операцию. # Выполнение лабораторной работы

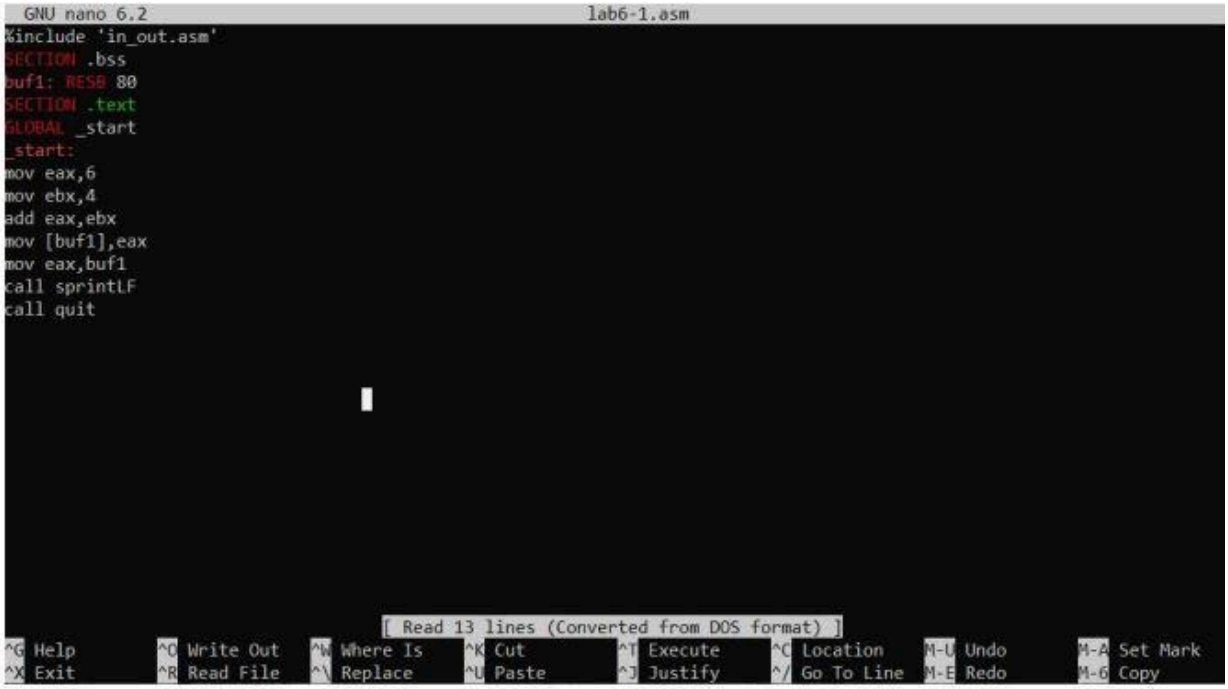
Создала каталог для программам лабораторной работы№6, перешлавнего и создала файл lab6-1.asm: mkdir ~/work/arch-pc/lab06 cd ~/work/arch-pc/lab06 touch lab6-1.asm

```
This message is shown once a day. To disable it please create the
/root/.hushlogin file.
root@LAPTOP-KE7VHG2Q:~# mkdir ~/work/arch-pc/lab06
root@LAPTOP-KE7VHG2Q:~# cd ~/work/arch-pc/lab06
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# touch lab06-1.asm
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#
```

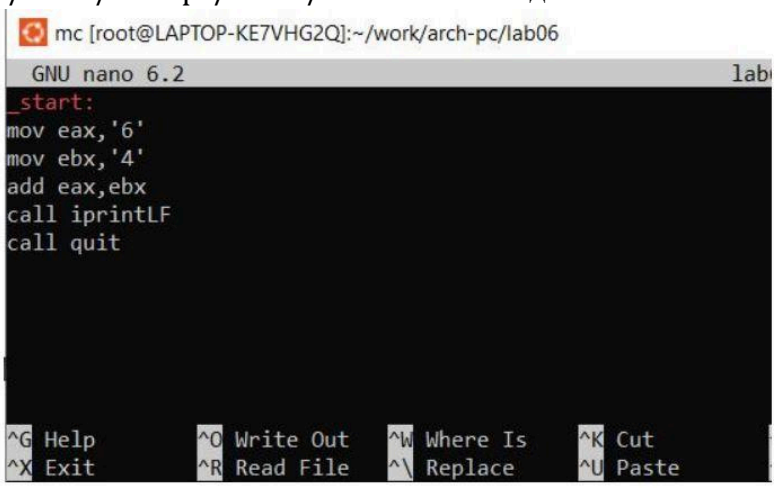
2. Рассмотрела примеры программ вывода символьныхичисленныхзначений. Программы будут выводить значения записанныеврегистреax. Затем ввела в файл lab6-1.asm текст программыиз листинга6.1.Вданной программе в регистр eax записала символ 6 (moveax,‘6’),врегистр ebx символ 4 (mov ebx,‘4’). Далее к значению в регистре eax прибавила значениерегистраebx (add eax,ebx, результат сложения запишется в регистреax). Потом вывела результат. Записала значение регистраeaxвпеременную buf1 (mov [buf1],eax), а потом адрес переменнойbuf1врегистр eax (mov eax,buf1) и вызовем функциюsprintLF. Ввела программу вывода значения регистра eax, создалаисполняемый файл и запустила его. nasm -f elf lab6-1.asm ld -m elf_i386 -o lab6-1 lab6-1.o ./lab6-1

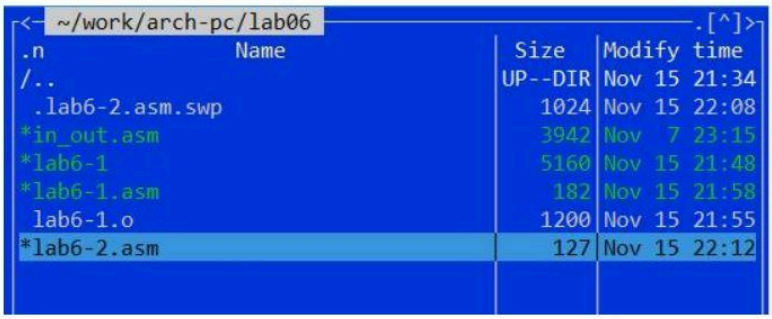


После изменила текст программы и вместо символов, записалаврегистры числа. Исправила текст программы (Листинг 6.1) следующимобразом: заменила строки mov eax,‘6’ mov ebx,‘4’ на строки mov eax,6 mov ebx,4



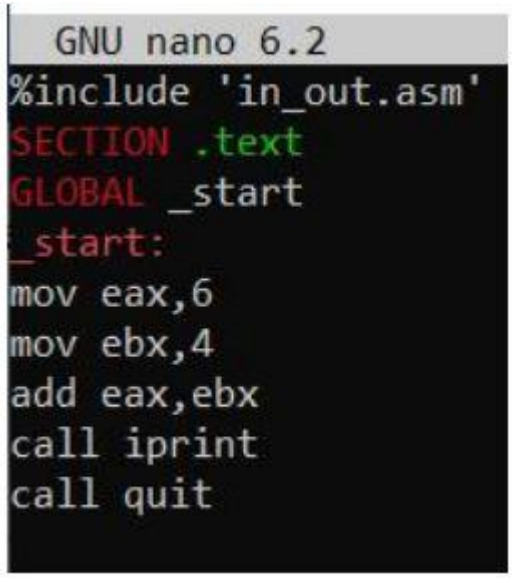
Создала исполняемый файл и запустила его. Преобразовала текст программы из Листинга 6.1 с использованием этих функций. Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2. touch ~/work/arch-pc/lab06/lab6-2.asm Создала исполняемый файл и запустила его.



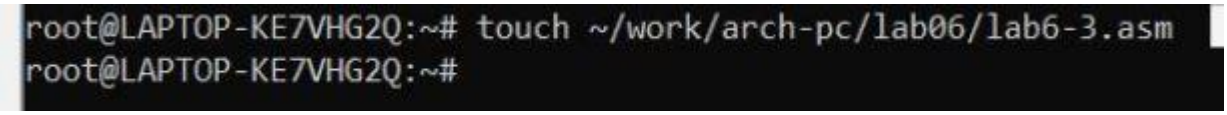


В результате работы программы я получила число 106 Аналогично предыдущему примеру изменила символы на числа. Потом заменила строки mov eax,‘6’ mov ebx,‘4’ на строки mov eax,6 mov ebx,4

Создала исполняемый файл и запустила его. Заменила функцию iprintLF на iprint, создала исполняемый файл и запустила его.



Создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06: touch ~/work/arch-pc/lab06/lab6-3.asm



Изучила текст программы из листинга 6.3 и ввела в lab6-3.asm. Создала исполняемый файл и запустила его. Результат работы программы должен быть следующим: user@dk4n31:~$ ./lab6-3

Результат: 4 Остаток от деления: 1 user@dk4n31:~$ Изменила текст программы для вычисления выражения

```
 GNU nano 6.2                                  lab6-3.asm
%include 'in_out.asm'
section .data
    result db 'Result: %d', 10, 0  ; Формат строки для вывода

section .bss
    res resd 1                     ; Резервируем место для результата

section .text
    global _start

_start:
    ; Вычисляем f(x) = (4 * 6 + 2) / 5
    mov eax, 4                     ; Загружаем 4 в регистр EAX
    imul eax, 6                    ; Умножаем EAX на 6 (EAX = 24)
    add eax, 2                     ; Добавляем 2 (EAX = 26)
    mov ebx, 5                     ; Загружаем 5 в регистр EBX
    xor edx, edx                   ; Обнуляем EDX перед делением
    div ebx                        ; Делим EAX на EBX (EAX = 26 / 5 = 5, EDX = 1)

    ; Сохраняем результат
    mov [res], eax                 ; Сохраняем результат в переменной res

    ; Выводим результат
    push dword [res]               ; Параметр для printf
    push result                    ; Строка формата
    call iprint                    ; Вызов функции printf
    add esp, 8                     ; Очистка стека

    ; Завершение программы
    mov eax, 1                     ; Системный вызов для выхода
    xor ebx, ebx                   ; Код возврата 0
    int 0x80                       ; Вызов ядра
```

```
                [ Read 32 lines (Converted from DOS format) ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify   ^/ Go To Line M-E Redo
```

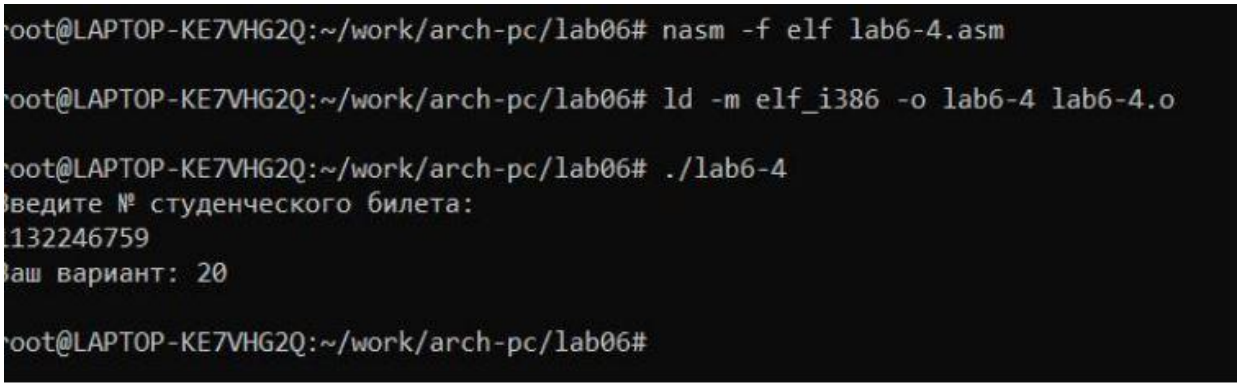
$f(x) = (4 * 6 + 2)/5$. Создала исполняемый файл и проверила его работу. Ответ: 5!

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ./lab6-3
5
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#
```

f(x) = (4 ∗ 6 + 2)/5

Создала исполняемый файл и проверила его работу. Ответ: 5!

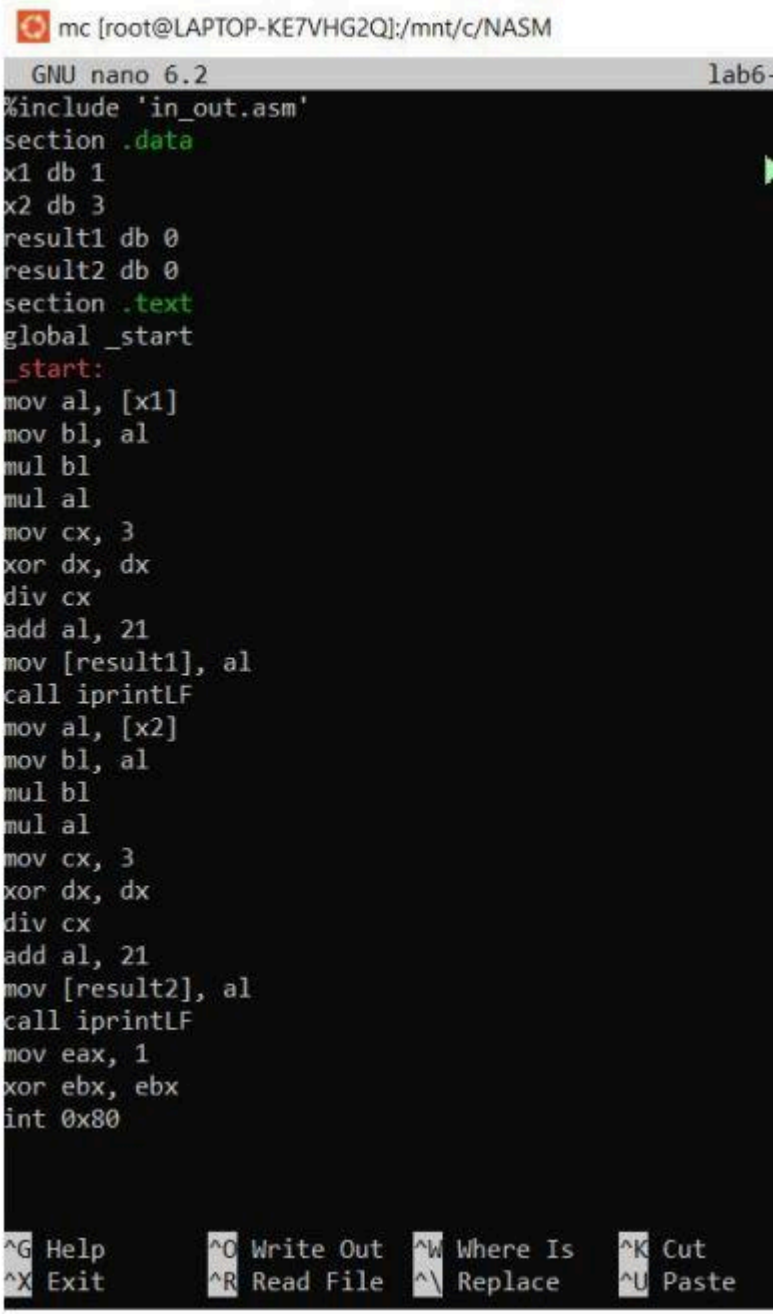
Далее вывела запрос на введение № студенческого билета, потом вычислила номер варианта по формуле:



(S mod 20) + 1, где S – номер студенческого билета(В данном случае a mod b – это остаток от деления a на b), потом вывела на экран номер варианта. Создала файл variant.asm в каталоге ~/work/arch-pc/lab06: touch ~/work/arch-pc/lab06/variant.asm Изучила текст программы из листинга 6.4 и ввела в файл variant.asm. Потом создала исполняемый файл и запустила его, далее проверила результат работы программы, вычислив номер варианта аналитически. Ответы на вопросы: 1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’? 20 2. Для чего используется следующие инструкции? mov ecx, x помещаем в регистр ecx число x mov edx, 80 длина строки 80 символов call sread считывает из строки длиной 80 символов значение переменной x3. Для чего используется инструкция “call atoi”? 4. Какие строки листинга 6.4 отвечают за вычисления варианта? 12 5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”? При этом остаток от целочисленного деления помещается в регистр AH, DX или EDX соответственно. 6. Для чего используется инструкция “inc edx”? -Инструкция inc edx используется для увеличения значения регистра EDX на 17. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? mov eax,edx call iprintLF

Выполняя самостоятельную работу я написала программу вычисления выражения y = f(x).

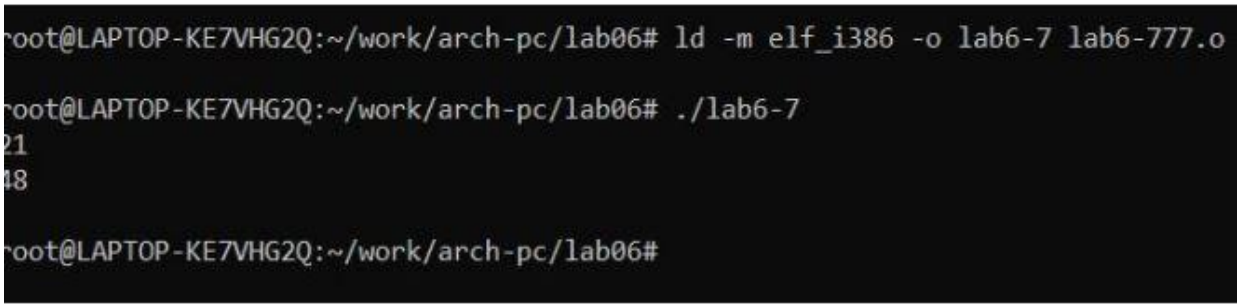
(Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x, вычислять заданное выражение в зависимости от введенного x, выводить



результат вычислений.)

Вид функции f(x) выбрала из таблицы 6.3 вариантов заданий в соответствии с номером, полученным при выполнении лабораторной работы.

Затем создала исполняемый файл и проверила его работу для значений x1 и x2 из 6.3. x1 = 1, ответ = 21 и x2 = 3, ответ = 48



Выводы

Целью было - узнать и ознакомиться с арифметическими инструкциями языка ассемблера NASM, проделав данные задания, я освоила инструкция в NASM.

Список литературы

1. GDB: The GNU Project Debugger. —URL: https://www.gnu.org/software/gdb/.2. GNU Bash Manual. — 2016. — URL: https://www.gnu.org/software/bash/manual/. 3. Midnight Commander Development Center. —2021. —URL: https://midnight-commander. org/. 4. NASM Assembly Language Tutorials. —2021. —URL: https://asmtutor.com/. 5. Newham C. Learning the bash Shell: Unix Shell Programming. —O'Reilly Media, 2005. — 354 с. — (In a Nutshell). —ISBN 0596009658. —URL: http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658.6. Robbins A. Bash Pocket Reference. —O'Reilly Media, 2016. —156с.— ISBN 978-1491941591. 7. The NASM documentation. — 2021. —URL: https://www.nasm.us/docs.php. 8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. —502с. —ISBN9781784396879. 9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. —М. : Форум, 2018.10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017. 11. Новожилов О. П. Архитектура ЭВМи систем. —М. : Юрайт, 2016.12. Расширенный ассемблер: NASM. —2021. —URL: https://www.opennet.ru/docs/RUS/nasm/. 13. Робачевский А., Немнюгин С., Стесик О. ОперационнаясистемаUNIX. — 2-е изд. — БХВПетербург, 2010. —656 с. —ISBN978-5-94157-538-1.14. Столяров А. Программирование на языке ассемблера NASMдляОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. —URL: http://www.stolyarov.info/books/asm_unix. 15. Таненбаум Э. Архитектура компьютера. —6-е изд. —СПб. : Питер,2013. — 874 с. — (Классика Computer Science). 16. Таненбаум Э., Бос Х. Современные операционные системы. —4-еизд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science