Course : Computer Organization – ENCM 369

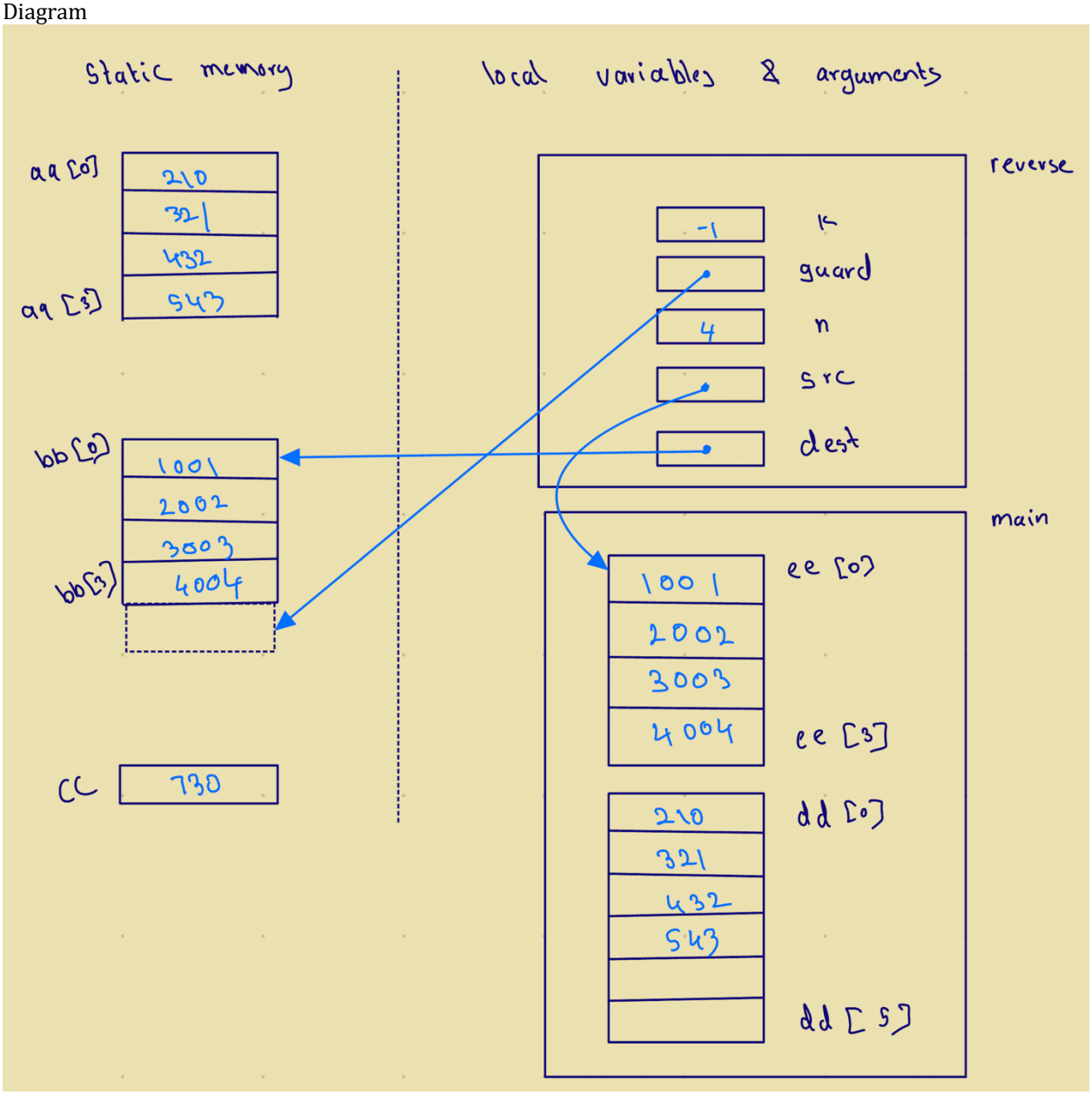
Lab # : Lab 1

Student Name : Nimna Wijedasa

Lab Section : B04

Ex B

Diagram


Static memory
local variables & arguments
aa [0]
210
321
432
aa [3]
543
bb [0]
1001
2002
3003
bb [3]
4004
cc
730
reverse
-1
k
guard
4
n
src
dest
main
1001
ee [0]
2002
3003
4004
ee [3]
210
dd [0]
321
432
543
dd [5]

Table

Instant in time	dest	Src	guard	k
Point 2 first	0x1000 7C	0x7ffe 40	0x1000 88	3
Point 2 second	0x1000 80	0x7ffe 40	0x1000 88	2
Point 2 third	0x1000 84	0x7ffe 40	0x1000 88	1
Point 2 last	0x1000 88	0x7ffe 40	0x1000 88	0

Ex G

Source code

```c
// lab1exG.c
// ENCM 369 Winter 2023 Lab 1 Exercise G

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_ABS_F (5.0e-9)
#define POLY_DEGREE 4

double polyval(const double *a, int n, double x);
/* Return a[0] + a[1] * x + ... + a[n] * pow(x, n). */

int main(void)
{
    double f[] = {1.47, 0.73, -2.97, -1.15, 1.00};
    double dfdx[POLY_DEGREE];

    double guess;
    int max_updates;
```

```
int update_count;
int n_scanned;
int i;

double current_x, current_f, current_dfdx;

printf("This program demonstrates use of Newton's Method to find\n"
       "approximate roots of the polynomial\nf(x) = ");
printf("%.2f", f[0]);


i = 1;
loop_top_1:
if (!(i <= POLY_DEGREE)) goto past_end_0;

    if (f[i] >= 0) goto else_code_1;
    printf(" - %.2f*pow(x,%d)", -f[i], i);
    goto end_if_1;
    else_code_1:
        printf(" + %.2f*pow(x,%d)", f[i], i);
        end_if_1:
i++;
goto loop_top_1;
past_end_0:

printf("\nPlease enter a guess at a root, and a maximum number of\n"
       "updates to do, separated by a space.\n");
n_scanned = scanf("%lf%d", &guess, &max_updates);

if (!(n_scanned != 2)) goto end_1;
printf("Sorry, I couldn't understand the input.\n");
exit(1);
end_1:
;

if (!(max_updates < 1)) goto end_2;
printf("Sorry, I must be allowed do at least one update.\n");
exit(1);
end_2:
;
printf("Running with initial guess %f.\n", guess);




for (i = POLY_DEGREE - 1; i >= 0; i--)
    dfdx[i] = (i + 1) * f[i + 1];   // Calculus!
```

```
    current_x = guess;
    update_count = 0;

    int k = 1;
    loop_top_in:
    if (k == 0) goto past_end_1;
        current_f = polyval(f, POLY_DEGREE, current_x);
        printf("%d update(s) done; x is %.15f; f(x) is %.15e\n",
               update_count, current_x, current_f);

        if (fabs(current_f) < MAX_ABS_F)
            goto past_end_1;
        if (update_count == max_updates)
            goto past_end_1;

        current_dfdx = polyval(dfdx, POLY_DEGREE - 1, current_x);
        current_x -= current_f / current_dfdx;
        update_count++;
        goto loop_top_in;
    past_end_1:

    if (fabs(current_f) >= MAX_ABS_F) goto else_code_2;
        printf("Stopped with approximate solution of %.10f.\n",
               current_x);
    goto end_if_5;

    else_code_2:
    printf("%d updates performed, |f(x)| still >= %g.\n",
           update_count, MAX_ABS_F);
    end_if_5:
    return 0;
}



double polyval(const double *a, int n, double x)
{
    double result = a[n];
    int i;
    i = n - 1;
    loop_top_2:
    if (!(i >= 0)) goto past_end_2;
    result *= x;
    result += a[i];
    i--;
    goto loop_top_2;
    past_end_2:
    return result;
```

```
}
```

Test runs

Left is goto C code that was converted from the C code which is on the right

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
1.0 x
Sorry, I couldn't understand the input.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
1.0 x
Sorry, I couldn't understand the input.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
1.0 0
Sorry, I must be allowed do at least one update.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
1.0 0
Sorry, I must be allowed do at least one update.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
0.8 10
Running with initial guess 0.800000.
0 update(s) done; x is 0.800000000000000; f(x) is -2.600000000000025e-02
1 update(s) done; x is 0.793782879005261; f(x) is -7.354453871588618e-05
2 update(s) done; x is 0.793765192642705; f(x) is -6.031051213994942e-10
Stopped with approximate solution of 0.7937651926.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
0.8 10
Running with initial guess 0.800000.
0 update(s) done; x is 0.800000000000000; f(x) is -2.600000000000025e-02
1 update(s) done; x is 0.793782879005261; f(x) is -7.354453871588618e-05
2 update(s) done; x is 0.793765192642705; f(x) is -6.031051213994942e-10
Stopped with approximate solution of 0.7937651926.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
2.2 10
Running with initial guess 2.200000.
0 update(s) done; x is 2.200000000000000; f(x) is -1.183999999999967e-01
1 update(s) done; x is 2.208734139864267; f(x) is 1.414853384897219e-03
2 update(s) done; x is 2.208632209347849; f(x) is 1.940833085356530e-07
3 update(s) done; x is 2.208632195361635; f(x) is 1.110223024625157e-15
Stopped with approximate solution of 2.2086321954.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
2.2 10
Running with initial guess 2.200000.
0 update(s) done; x is 2.200000000000000; f(x) is -1.183999999999967e-01
1 update(s) done; x is 2.208734139864267; f(x) is 1.414853384897219e-03
2 update(s) done; x is 2.208632209347849; f(x) is 1.940833085356530e-07
3 update(s) done; x is 2.208632195361635; f(x) is 1.110223024625157e-15
Stopped with approximate solution of 2.2086321954.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-1.07 10
Running with initial guess -1.070000.
0 update(s) done; x is -1.070000000000000; f(x) is 8.142459999999518e-03
1 update(s) done; x is -1.065384817690193; f(x) is 1.611521383877434e-04
2 update(s) done; x is -1.065289717816483; f(x) is 6.796855567259286e-08
3 update(s) done; x is -1.065289677672691; f(x) is 1.243449787580175e-14
Stopped with approximate solution of -1.0652896777.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-1.07 10
Running with initial guess -1.070000.
0 update(s) done; x is -1.070000000000000; f(x) is 8.142459999999518e-03
1 update(s) done; x is -1.065384817690193; f(x) is 1.611521383877434e-04
2 update(s) done; x is -1.065289717816483; f(x) is 6.796855567259286e-08
3 update(s) done; x is -1.065289677672691; f(x) is 1.243449787580175e-14
Stopped with approximate solution of -1.0652896777.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-0.8 10
Running with initial guess -0.800000.
0 update(s) done; x is -0.800000000000000; f(x) is -1.640000000000019e-02
1 update(s) done; x is -0.786623164763458; f(x) is 6.391708412911701e-04
2 update(s) done; x is -0.787107095386452; f(x) is 8.099621247037447e-07
3 update(s) done; x is -0.787107710185623; f(x) is 1.308952946033060e-12
Stopped with approximate solution of -0.7871077102.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-0.8 10
Running with initial guess -0.800000.
0 update(s) done; x is -0.800000000000000; f(x) is -1.640000000000019e-02
1 update(s) done; x is -0.786623164763458; f(x) is 6.391708412911701e-04
2 update(s) done; x is -0.787107095386452; f(x) is 8.099621247037447e-07
3 update(s) done; x is -0.787107710185623; f(x) is 1.308952946033060e-12
Stopped with approximate solution of -0.7871077102.
nimnawijedasa@DaMacBook lab1 %
```

```
nimnawijedasa@DaMacBook exG % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-4.8 10
Running with initial guess -4.800000.
0 update(s) done; x is -4.800000000000000; f(x) is 5.875595999999999e+02
1 update(s) done; x is -3.607261669380082; f(x) is 1.834907056922019e+02
2 update(s) done; x is -2.735534758422163; f(x) is 5.678662379243129e+01
3 update(s) done; x is -2.109576217183543; f(x) is 1.731437128945177e+01
4 update(s) done; x is -1.672849511438880; f(x) is 5.152227785778270e+00
5 update(s) done; x is -1.381980454936464; f(x) is 1.471757193904503e+00
6 update(s) done; x is -1.202666238588805; f(x) is 3.887934620347735e-01
7 update(s) done; x is -1.107243653267704; f(x) is 8.466227950872973e-02
8 update(s) done; x is -1.071255513598581; f(x) is 1.036951015902532e-02
9 update(s) done; x is -1.065440933827905; f(x) is 2.562677900719290e-04
10 update(s) done; x is -1.065289779095831; f(x) is 1.717223301334059e-07
10 updates performed, |f(x)| still >= 5e-09.
nimnawijedasa@DaMacBook exG %
```

```
nimnawijedasa@DaMacBook lab1 % ./a.out
This program demonstrates use of Newton's Method to find
approximate roots of the polynomial
f(x) = 1.47 + 0.73*pow(x,1) - 2.97*pow(x,2) - 1.15*pow(x,3) + 1.00*pow(x,4)
Please enter a guess at a root, and a maximum number of
updates to do, separated by a space.
-4.8 10
Running with initial guess -4.800000.
0 update(s) done; x is -4.800000000000000; f(x) is 5.875595999999999e+02
1 update(s) done; x is -3.607261669380082; f(x) is 1.834907056922019e+02
2 update(s) done; x is -2.735534758422163; f(x) is 5.678662379243129e+01
3 update(s) done; x is -2.109576217183543; f(x) is 1.731437128945177e+01
4 update(s) done; x is -1.672849511438880; f(x) is 5.152227785778270e+00
5 update(s) done; x is -1.381980454936464; f(x) is 1.471757193904503e+00
6 update(s) done; x is -1.202666238588805; f(x) is 3.887934620347735e-01
7 update(s) done; x is -1.107243653267704; f(x) is 8.466227950872973e-02
8 update(s) done; x is -1.071255513598581; f(x) is 1.036951015902532e-02
9 update(s) done; x is -1.065440933827905; f(x) is 2.562677900719290e-04
10 update(s) done; x is -1.065289779095831; f(x) is 1.717223301334059e-07
10 updates performed, |f(x)| still >= 5e-09.
nimnawijedasa@DaMacBook lab1 %
```

Ex H

```c
// lab1exH.c
// ENCM 369 Winter 2023 Lab 1 Exercise H

#include <stdio.h>

void print_array(const char *str, const int *a, int n);
// Prints the string given by str on stdout, then
// prints a[0], a[1], ..., a[n - 1] on stdout on a single line.

void sort_array(int *x, int n);
// Sorts x[0], x[1], ..., x[n - 1] from smallest to largest.

int main(void)
{
  int test_array[] = { 4000, 5000, 7000, 1000, 3000, 4000, 2000, 6000 };

  print_array("before sorting ...", test_array, 8);
  sort_array(test_array, 8);
  print_array("after sorting ...", test_array, 8);
  return 0;
}

void print_array(const char *str, const int *a, int n)
{
    int i = 0;
    puts(str);
    start:
        if (!(i < n)) goto end;
        printf("    %d", a[i]);
        printf("\n");
        i++;
        goto start;
    end:
    ;
};

void sort_array(int *x, int n)
{
  // This is an implementation of an algorithm called insertion sort.

  int outer = 1, inner, vti;
    outer_loop:
        if (!(outer < n)) goto quit_outer_loop;
```

```
        vti = x[outer];
        inner = outer;

    inner_loop:
        if (!(inner > 0)) goto quit_inner_loop;
        if (!(vti < x[inner - 1])) goto quit_inner_loop;
        x[inner] = x[inner - 1];
        inner--;
    goto inner_loop;
    quit_inner_loop:
    x[inner] = vti;
    outer++;
    goto outer_loop;
    quit_outer_loop:;
};
```

```
before sorting ...
    4000
    5000
    7000
    1000
    3000
    4000
    2000
    6000
after sorting ...
    1000
    2000
    3000
    4000
    4000
    5000
    6000
    7000
Program ended with exit code: 0
```