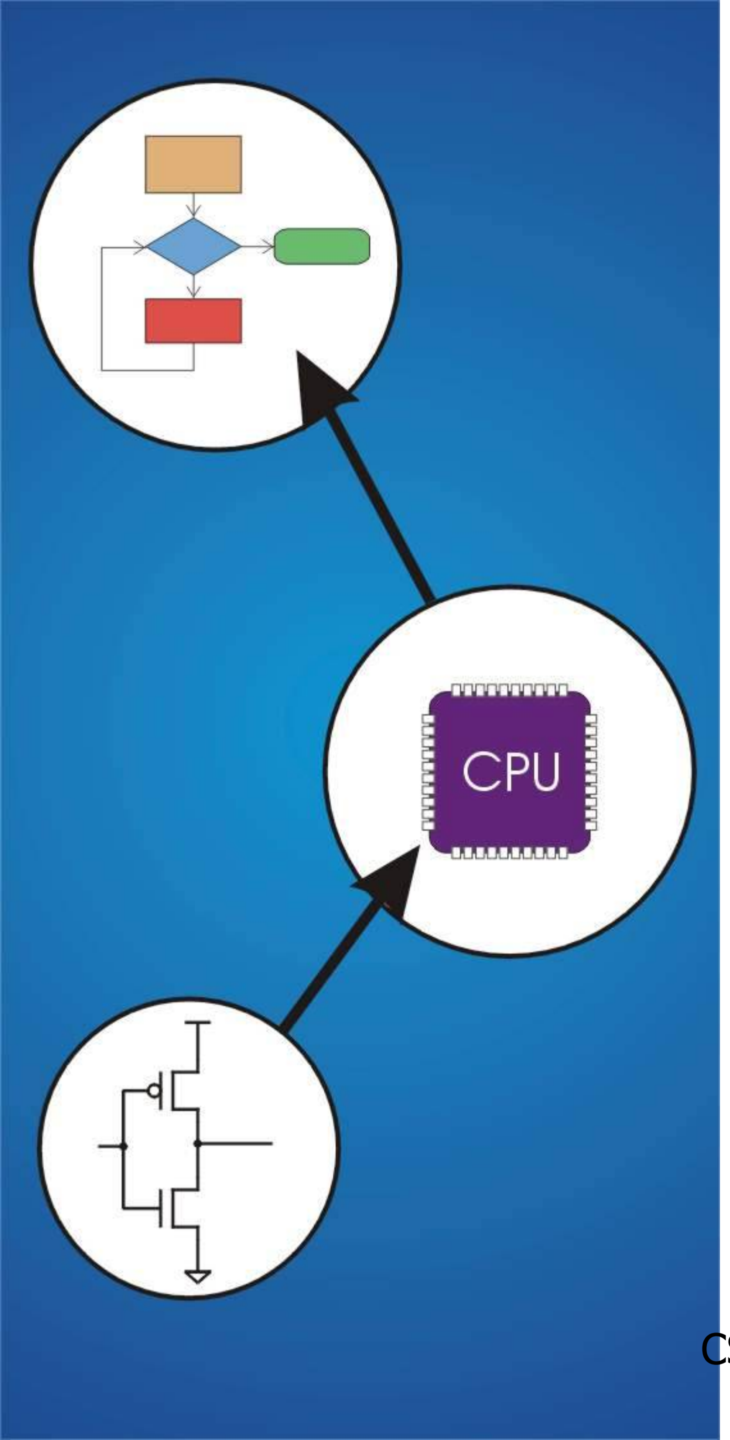


Midterm 1
Review Slides

Review Topics

- Binary Representation:
  - binary numbers, signed int, floating point
  - ASCII
  - Bitwise operations
- C operators, structures
- Pointers, * and &, arrays, struct, typedef
- Dynamic memory allocation, linked lists
- Functions, Stack-frames, recursion
- I/O: stdin/stdout, formatting, file

Converting Decimal to Binary (2's C)

First Method: *Division*

1. **Find magnitude of decimal number. (Always positive.)**
2. **Divide by two – remainder is least significant bit.**
3. **Keep dividing by two until answer is zero,**
   writing remainders from right to left.
4. **Append a zero as the MS bit;**
   if original number was negative, take two's complement.

$X = 104_{ten}$

104/2	=	52 r0	*bit 0*
52/2	=	26 r0	*bit 1*
26/2	=	13 r0	*bit 2*
13/2	=	6 r1	*bit 3*
6/2	=	3 r0	*bit 4*
3/2	=	1 r1	*bit 5*
1/2	=	0 r1	*bit 6*

$X = 01101000_{two}$

Converting Decimal to Binary (2's C)

Second Method: *Subtract Powers of Two*

1. **Find magnitude of decimal number.**
2. **Subtract largest power of two less than or equal to number.**
3. **Put a one in the corresponding bit position.**
4. **Keep subtracting until result is zero.**
5. **Append a zero as MS bit; if original was negative, take two's complement.**

n	2^n
0	1
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024

$X = 104_{ten}$

104 - 64	=	40		*bit 6*
40 - 32	=	8		*bit 5*
8 - 8	=	0		*bit 3*

$X = 01101000_{two}$

Number Representation
Binary to Hexadecimal Conversion

- Method: Group binary digits, convert to hex digits using table.
- Question: What is binary 01100111100010011111111011011010 in hexadecimal?

```
0110 0111 1000 1001 1111 1110 1101 1010
6    7    8    9    F    E    D    A
```

- Answer: `0x6789FEDA`

Hexadecimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Number Representation
Two's Complement

- Invert all the bits, and add 1.
- Question: What is the value -8 in (16-bit) 2's complement form?

```
8 = 0x0008 = 0000 0000 0000 1000
Invert bits  1111 1111 1111 0111
Add one    + 0000 0000 0000 0001
Answer     = 1111111111111000 (binary)
Answer     = 0xFFF8
```

Answer: `0xFFF8 = -8 decimal`

Two's Complement

Two's complement **representation developed to make circuits easy for arithmetic.**

- **for each positive number (X), assign value to its negative (-X), such that X + (-X) = 0 with "normal" addition, ignoring carry out**

	00101	(5)		**01001**	(9)
+	**11011**	(-5)	+	______	(-9)
	00000	(0)		**00000**	(0)

Number Representation
Binary to Floating Point Conversion

- Single-precision IEEE floating point number:

$$\underbrace{1}_{\textit{sign}}\ \underbrace{01111110}_{\textit{exponent}}\ \underbrace{10000000000000000000000}_{\textit{fraction}}$$

  - Or 0xBF400000
  - Sign is 1 – number is negative.
  - Exponent field is 01111110 = 126 – 127 = -1 (decimal).
  - Fraction is 1.100000000000… = 1.5 (decimal).
- Value = $-1.5 \times 2^{(126-127)} = -1.5 \times 2^{-1} = -0.75$

Floating Point to Binary Conversion

- Value = 4.25
  - Number is positive – sign is 0
  - Fraction is 100.01 (binary), normalize to 1.0001 * 2^2
  - Exponent is 2 + 127 = 129 (decimal) = 10000001
- Single-precision IEEE floating point number:

$$\underbrace{0}_{\textit{sign}}\ \underbrace{10000001}_{\textit{exponent}}\ \underbrace{00010000000000000000000}_{\textit{fraction}}$$

- or 0x40880000

Operators

- arithmetic operators:

```
int i = 10;
int j = 2;
printf("d\n", i + j); 12
printf("d\n", i - j); 8
printf("d\n", i * j); 20
printf("d\n", i / j); 5
```

- bitwise operators:

```
int i = 0x11223344;
int j = 0xFFFF0000;
printf("0x%x\n", ~i); 0xEEDDCCBB
printf("0x%x\n", i & j); 0x11220000
printf("0x%x\n", i | j); 0xFFFF3344
```

- logical operators: (in C 0 is false, everything else is true!)

```
int i = 0x11223344;
int j = 0x00000000;
printf("0x%x\n", !i); 0x00000000
printf("0x%x\n", i && j); 0x00000000
printf("0x%x\n", i || j); 0x00000001
```

C Programming

Control Structures

- C conditional and iterative statements
  - if statement

```
if (value == 0x12345678)
    printf("value matches 0x12345678\n");
```

  - for loop

```
for (int i = 0; i < 8; ++i)
    printf("i = %d\n", i);
```

  - while loop

```
int j = 6;
while (j--)
    printf("j = %d\n", j);
```

Functions in C

```
double ValueInDollars(double amount, double rate);
```

Declaration (Prototype)

```
main()
{
   ...
```

function call (invocation)

```
   dollars = ValueInDollars(francs,
                            DOLLARS_PER_FRANC);
   printf("%f francs equals %f dollars.\n",
          francs, dollars);
   ...
}
```

definition (function code)

```
double ValueInDollars(double amount, double rate)
{
   return amount * rate;
}
```

Pointers: * and & operators

```
int i;
int *ptr;

i = 4;
ptr = &i;
*ptr = *ptr + 1;
```

store the value 4 into the memory location associated with i

store the address of i into the memory location associated with ptr

read the contents of memory at the address stored in ptr

store the result into memory at the address stored in ptr

Relationship between Arrays and Pointers

An array name is essentially a pointer to the first element in the array

```
char word[10];
char *cptr;
cptr = word;  /* points to word[0] */
```

Difference:

Can change the contents of cptr, as in

```
cptr = cptr + 1;
```

(The identifier "word" is not a variable.)

Correspondence:

`cptr`	`word`	`&word[0]`
`(cptr + n)`	`word + n`	`&word[n]`
`*cptr`	`*word`	`word[0]`
`*(cptr + n)`	`*(word + n)`	`word[n]`

C Programming
Pointers and Arrays

- C pointers and arrays

```
void foo(int *pointer)
{
     *(pointer+0) = pointer[2] = 0x1234;
     *(pointer+1) = pointer[3] = 0x5678;
}
int main(int argc, char *argv[])
{
   int array[]= {0, 1, 2, 3};
   foo(array);
   for (int i = 0; i <= 3; ++i)
      printf(“array[%d] = %x\n”, i, array[i]);
}
```

C Programming
Strings

- C strings

```
char *string = “hello”;
char *carray = { ‘h’,’e’,’l’,’l’,’o’ };
char label[20];
strcpy(label, “hello”);
strcat(label, “ world”);
printf(“%s\n”, label); hello world
printf(“%d\n”, strlen(label)); 11
printf(“%d\n”, sizeof(label)); 20
```

C Programming
Data Structures

- C data structures

```
// Structure definition
struct sCoordinate
{
    float X;
    float y;
    float z;
};
typedef struct {
    …
} Coordinate;
```

C Programming
Data Structures

- C data structures

```
// Structure allocation
struct sCoordinate coordinates[10]; // no typedef
Coordinate coordinates[10]; // typedef
Coordinate *coordinates =
           malloc(sizeof(Coordinate)*10);

// Structure access
coordinates[5].X = 1.0f;
pCoordinate->X = 1.0f;
```

Dynamic memory Allocation

```
planes = (Flight*) malloc(n* sizeof(Flight));


..
newNode->next = nextNode;
            means (*newNode).next =nextNode
..

free(newNode);
```

Scanning the linked List of structs

```
Car *ScanList(Car *head, int searchID)
{
  Car *previous, *current;
  previous = head;
  current = head->next;
  /* Traverse until ID >= searchID */
  while ((current!=NULL)
          && (current->vehicleID < searchID)) {
    previous = current;
    current = current->next;
  }
  return previous;
}
```

printf / scanf

```
int a = 100;          int b = 65;        char c = 'z';
char banner[10] = "Hola!";      double pi = 3.14159;

printf("The variable 'a' decimal: %d\n", a);
printf("The variable 'a' hex: %x\n", a);
printf("The variable 'a' binary: %b\n", a);
printf("'a' plus 'b' as character: %c\n", a+b);
printf("A char %c.\t A string %s\n A float %f\n",
       c, banner, pi);
```

```
char name[100];     int bMonth, bDay, bYear;
double gpa;

scanf("%s %d/%d/%d %lf",
      name, &bMonth, &bDay, &bYear, &gpa);
```

File I/o

The type of a stream is a "file pointer", declared as:

```
FILE *infile;
```

The fopen function associates a physical file with a stream.

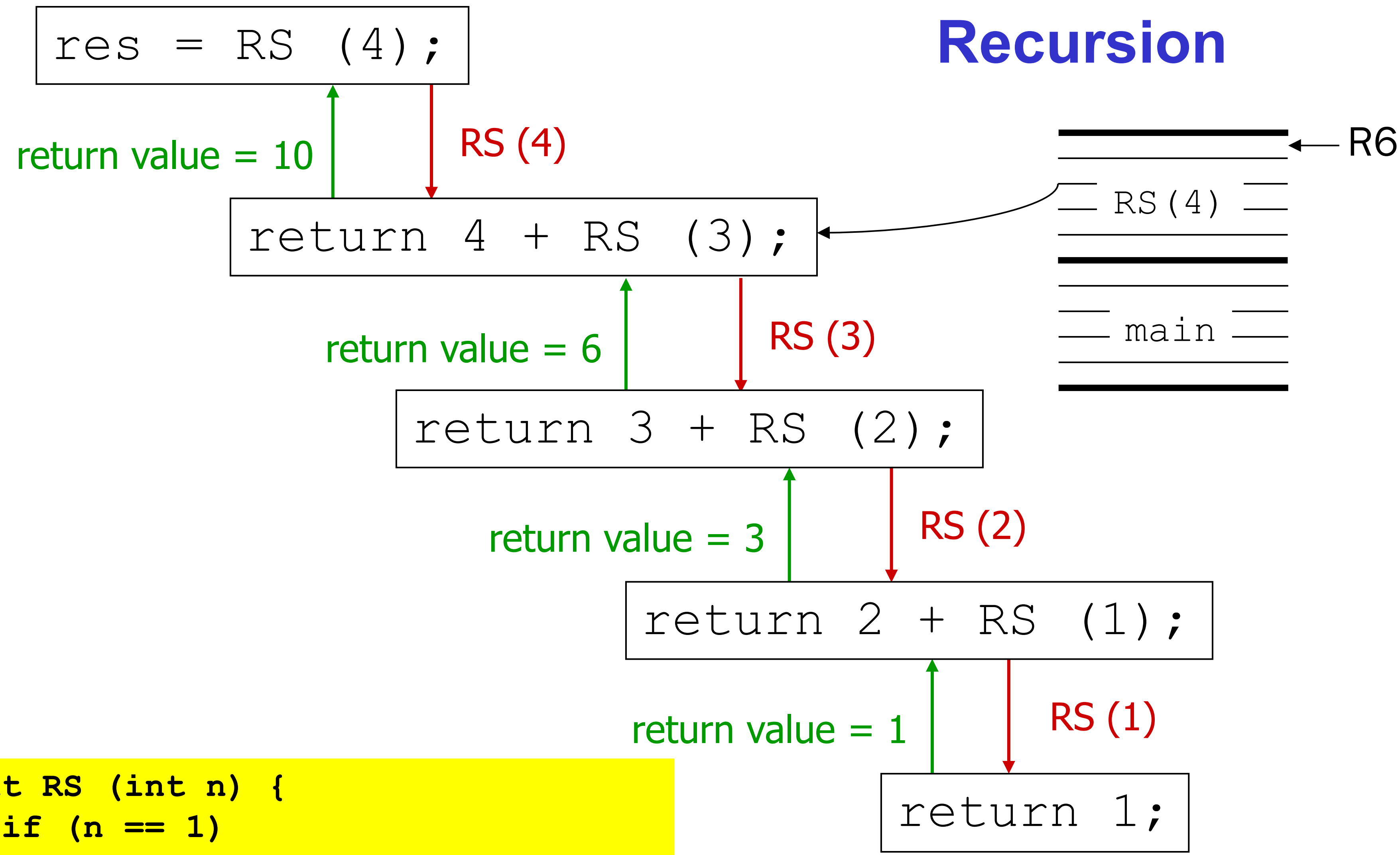
```
FILE *fopen(char* name, char* mode);
```

Once a file is opened, it can be read or written using `fscanf()` and `fprintf()`, respectively.

```
fprintf(outfile, "The answer is %d\n", x);

fscanf(infile, "%s %d/%d/%d %lf",
        name, &bMonth, &bDay, &bYear, &gpa);
```

Recursion




```
int RS (int n) {
  if (n == 1)
    return 1;
  else
    return n + RS (n-1);
}
```