

CS 3330 Introduction

Daniel and Charles

lecturers

- Charles and I will be splitting lectures same(ish) lecture in each section

Grading



Take Home Quizzes:
10% (10% dropped)



Midterms (2): 30%



Final Exam
(cumulative): 20%



Homework + Labs:
40%

late policy

- ❑ exceptional circumstance? contact us.
- ❑ otherwise, for **homework only**:
 - ❑ -10% 0 to 48 hours late
 - ❑ -15% 48 to 72 hours late
 - ❑ -100% otherwise
- ❑ late quizzes, labs: **no**
we release answers talk to us if illness,
etc.

Coursework

- quizzes — pre/post week of lecture
 - you will need to **read**
- labs — grading: did you make reasonable progress?
 - collaboration permitted
- homework assignments — introduced by lab (mostly)
 - due at 9am on the next lab day (mostly)
complete individually
- exams — multiple choice/short answer — 2 + final

Collaboration Policy

- You are encouraged to discuss homework and final project assignments with other students in the class, as long as the following rules are followed:

Collaboration Policy

You can't view other peoples code. That includes pseudo code.

You can discuss the assignment generally.

Sharing code in labs is allowed

Attendance?




Lecture: strongly recommended but not required. lectures are recorded to help you review



Lab: electronic, remote-possible submission, usually.

lecture/lab/HW
synchronization

labs/HWs not quite
synchronized with
lectures



main problem: want to
cover material **before you**
need it in lab/HW

Quizzes?

- linked off course website (First quiz, due 11 of September)
- pre-quiz, on reading – released by Saturday evening, due Tuesdays, 12:15 PM (Which is just before lecture)
- post-quiz, on lecture topics — released Thursday evening, due following Saturday, 11:59 PM
- each quiz 90 minute time limit (+ adjustments if SDAC says) lowest 10% (approx. 2 quizzes) will be dropped (Quizzes are multiple choice and normally about 5 questions)

TAs/Office Hours

- Office hours will be posted on the calendar on the website
- Still discussion hours with TAs.
- Office hours will start next week.



Your TODO list

- Quizzes!
 - post-quiz after Thursday lecture pre-quiz before Tuesday lecture
- lab account and/or C environment working
 - lab accounts should happen by this weekend
- before lab next week



Questions?

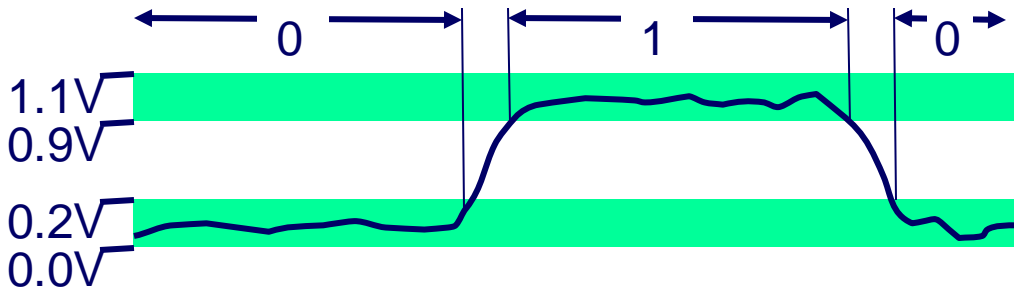


Let's Build a simple machine

How will store information in our machine?

Everything is bits

- Each bit is 0 or 1
- Why bits? Electronic Implementation
 - Reliably transmitted on noisy and inaccurate wires



There are different ways to represent bits

Encoding Byte Values

- Byte = 8 bits
 - Binary 00000000_2 to 11111111_2
 - Decimal: 0_{10} to 255_{10}
 - Hexadecimal 00_{16} to FF_{16}
 - Base 16 number representation
 - Use characters '0' to '9' and 'A' to 'F'
 - Write $FA1D37B_{16}$ in C as
 - $0xFA1D37B$
 - $0xfa1d37b$

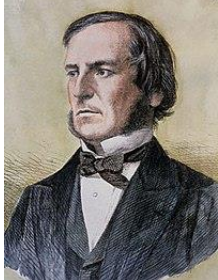
Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Q: $0x605C + 0x5 = 0x606$

Boolean Algebra

- Developed by George Boole in 19th Century
- Algebraic representation of logic
 - Encode “True” as 1 and “False” as 0
 - The symbols here are how you do these operation in c

Not an I



And

- $A \& B = 1$ when both $A=1$ and $B=1$

$\&$	0	1
0	0	0
1	0	1

Not

- $\sim A = 1$ when $A=0$

\sim	
0	1
1	0

Or

- $A | B = 1$ when either $A=1$ or $B=1$

	0	1
0	0	1
1	1	1

Exclusive-Or (Xor)

- $A \wedge B = 1$ when either $A=1$ or $B=1$, but not both

\wedge	0	1
0	0	1
1	1	0

Boolean Algebra

- Could we develop a machine that adds two one-bit numbers using any of these gates
 - Encode “True” as 1 and “False” as 0

And

- $A \& B = 1$ when both $A=1$ and $B=1$

$\&$	0	1
0	0	0
1	0	1

Not

- $\sim A = 1$ when $A=0$

\sim	
0	1
1	0

Or

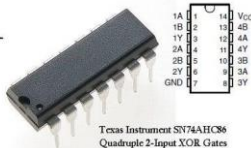
- $A | B = 1$ when either $A=1$ or $B=1$

	0	1
0	0	1
1	1	1

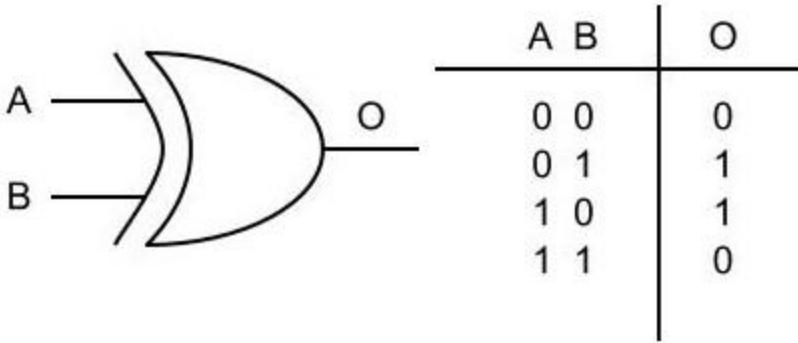
Exclusive-Or (Xor)

- $A \wedge B = 1$ when either $A=1$ or $B=1$, but not both

\wedge	0	1
0	0	1
1	1	0

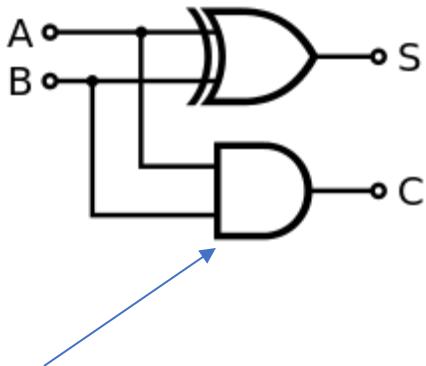


Simple One Bit Adder (Not Quite)



Suppose that we had extra place to hold that last result bit what gate could we use to find it?

Simple Half Adder (Not Quite)

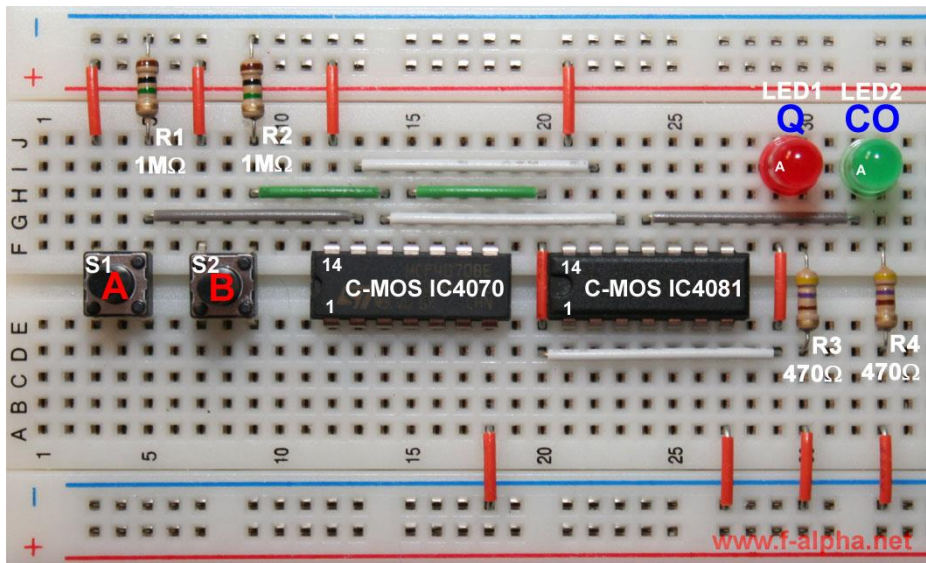


And Gate

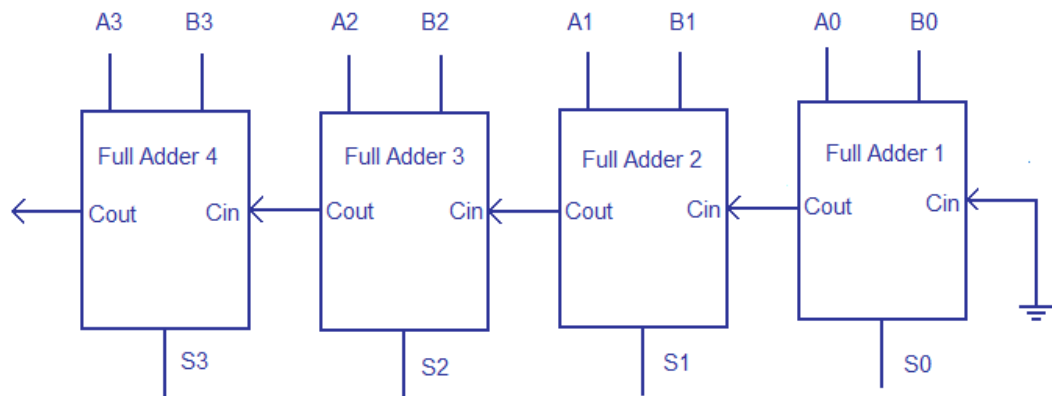
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Binary for 2

Half Adder Bread board



Ripple Carry Adder



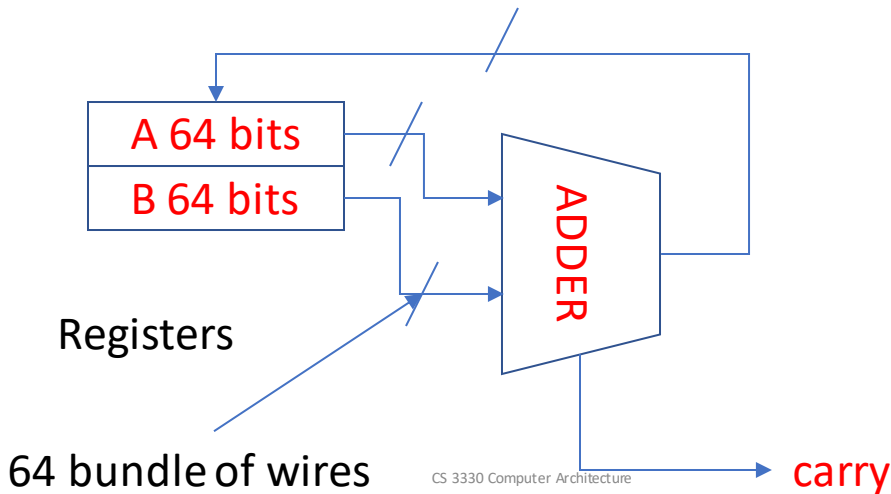
4 bit ripple carry adder

www.circuitstoday.com

<http://www.circuitstoday.com/wp-content/uploads/2012/03/ripple-carry-adder.png>

Now we have a machine that can add large numbers

(Bundle of wires)



How do we program it?

We could put one and zeros in manually

The solution: Abstraction

Layers of abstraction

“Higher-level” language: C

$x += y$

Assembly

`addq %rdi %rsi`

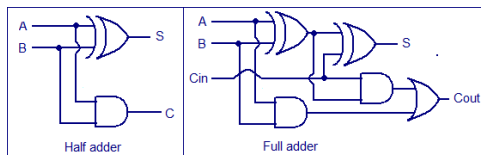
Machine code

0010 0001

Y86 64 bit simplified

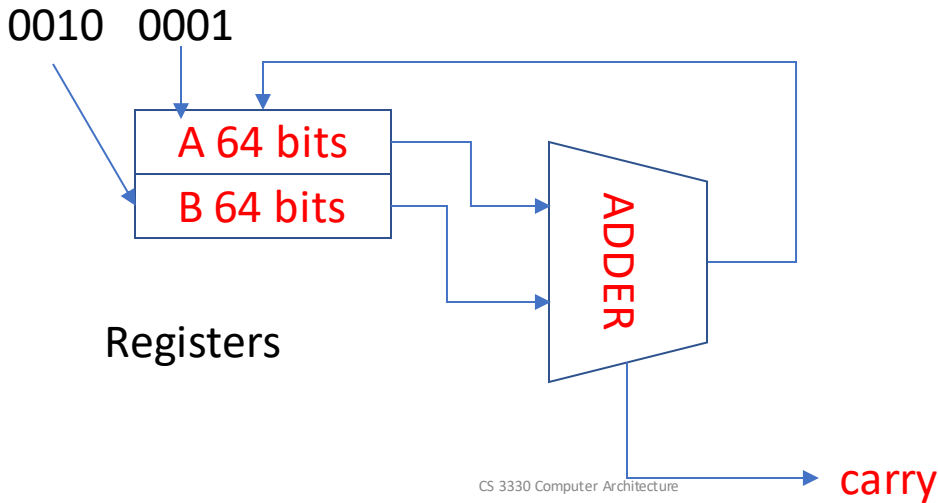
Hardware Design Language: HCLRS/VHDL

Gates / Transistors / Wires / Registers



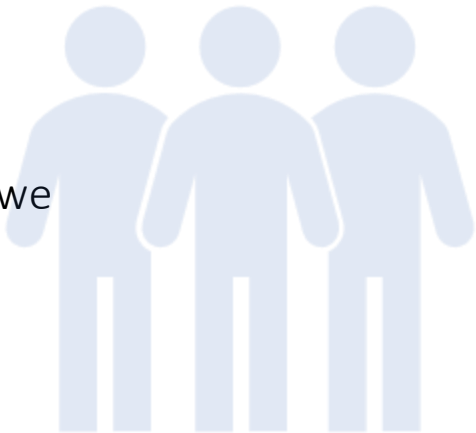
Now we have a machine
that can add large numbers

How do we
program it?





We are computer scientists why should we care about hardware?



Why

- Understanding computer architecture will help you:
 - Write fast programs
 - And understand strange program behaviors like segmentation faults.

Let's look at a simple example

Memory System Performance Example

```
void copyij(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (i = 0; i < 2048; i++)
        for (j = 0; j < 2048; j++)
            dst[i][j] = src[i][j];
}
```

4.3ms

```
void copyji(int src[2048][2048],
            int dst[2048][2048])
{
    int i,j;
    for (j = 0; j < 2048; j++)
        for (i = 0; i < 2048; i++)
            dst[i][j] = src[i][j];
}
```

81.8ms

2.0 GHz Intel Core i7 Haswell

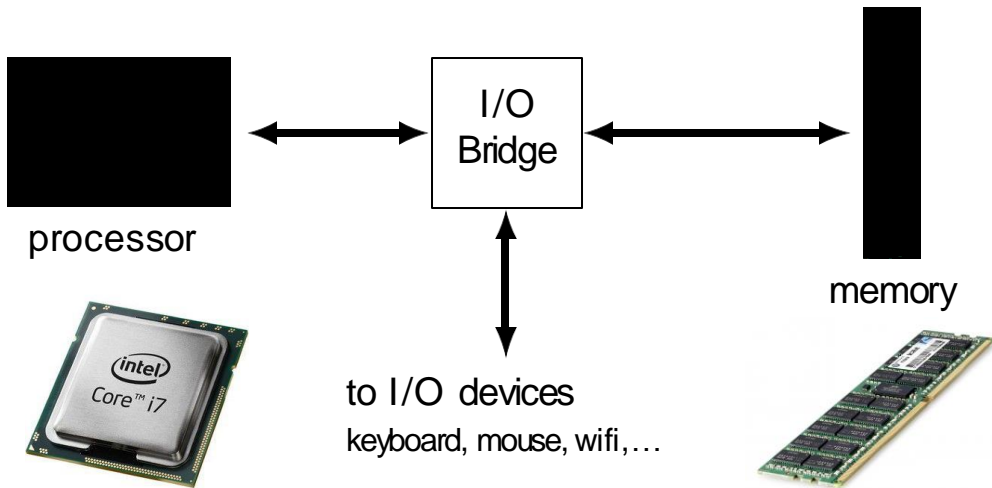
- Hierarchical memory organization
- Performance depends on access patterns
 - Including how step through multi-dimensional array

program performance: issues

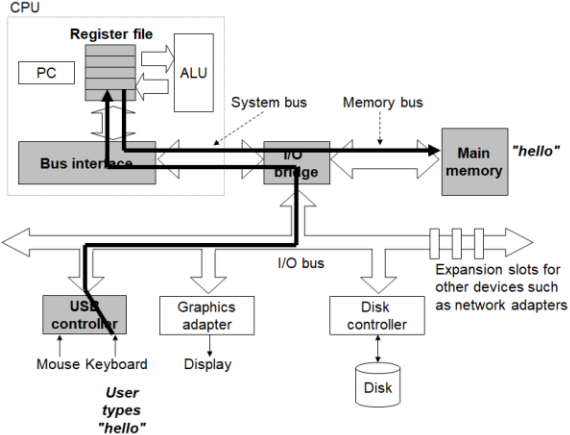
- (Hardware) Parallelism
 - How do we write program to take advantage of parallelism
- (Hardware) caching
 - accessing things recently accessed is faster
 - need reuse of data/code
- (Software) (more in other classes: algorithmic efficiency) (Time and Space Complexity Big O)

Let's start by looking at high-level overview of architecture of a system

processors and memory



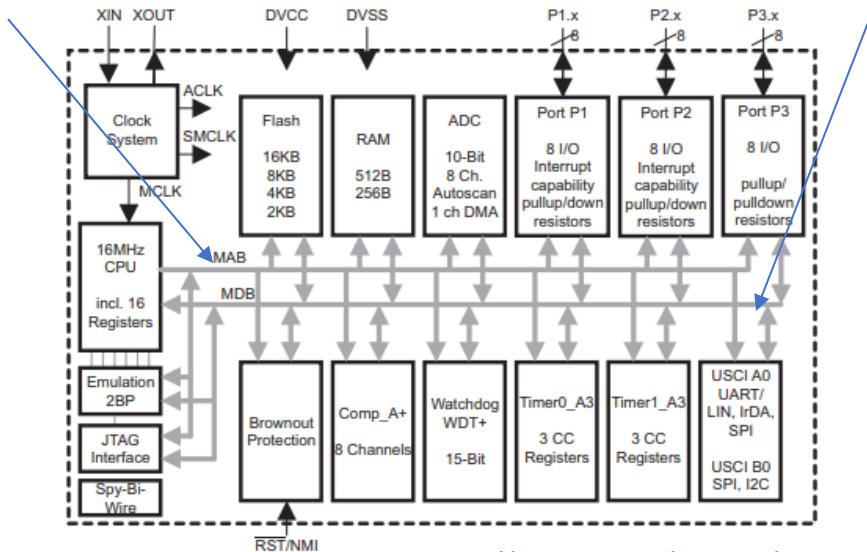
More detail



Memory Address bus
Get me value at
0x0003

Memory Data bus
Your data was:
0xffff

Functional Block Diagram, MSP430G2x53



Schematic

<http://www.ti.com/product/MSP430G2553>

Endianness

Endianness

address	value
0xFFFFFFFF	0x14
0xFFFFFFF0	0x45
0xFFFFFFF4	0xDE
...	...
0x00042006	0x06
0x00042005	0x05
0x00042004	0x04
0x00042003	0x03
0x00042002	0x02
0x00042001	0x01
0x00042000	0x00
0x00041FFF	0x03
0x00041FFE	0x60
...	...
0x00000002	0xFE
0x00000001	0xE0

```
int *x = (int*)0x42000;  
cout << *x << endl;
```

0x03020100 = 50462976

little endian
(least significant byte has lowest address)

0x00010203 = 66051

big endian
(most significant byte has lowest address)

memory

address	value
0xFFFFFFFF	0x14
0xFFFFFFF0	0x45
0xFFFFFFF4	0xDE
...	...
0x00042006	0x06
0x00042005	0x05
0x00042004	0x04
0x00042003	0x03
0x00042002	0x02
0x00042001	0x01
0x00042000	0x00
0x00041FFF	0x03
0x00041FFE	0x60
...	...
0x00000002	0xFE
0x00000001	0xE0
0x00000000	0xA0

address	value
0x00000000	0xA0
0x00000001	0xE0
0x00000002	0xFE
...	...
0x00041FFE	0x60
0x00041FFF	0x03
0x00042000	0x00
0x00042001	0x01
0x00042002	0x02
0x00042003	0x03
0x00042004	0x04
0x00042005	0x05
0x00042006	0x06
...	...
0xFFFFFFFF	0xDE
0xFFFFFFF0	0x45
0xFFFFFFF4	0x14

Endianness

address	value
0xFFFFFFFF	0x14
0xFFFFFFF0	0x45
0xFFFFFFF4	0xDE
...	...
0x00042006	0x06
0x00042005	0x05
0x00042004	0x04
0x00042003	0x03
0x00042002	0x02
0x00042001	0x01
0x00042000	0x00
0x00041FFF	0x03
0x00041FFE	0x60
...	...
0x00000002	0xFE
0x00000001	0xE0

$$0x03020100 = 50462976$$

$$0x03020101 = 50462977$$

little endian
(least significant byte has lowest address)

$$0x00010203 = 66051$$

big endian
(most significant byte has lowest address)

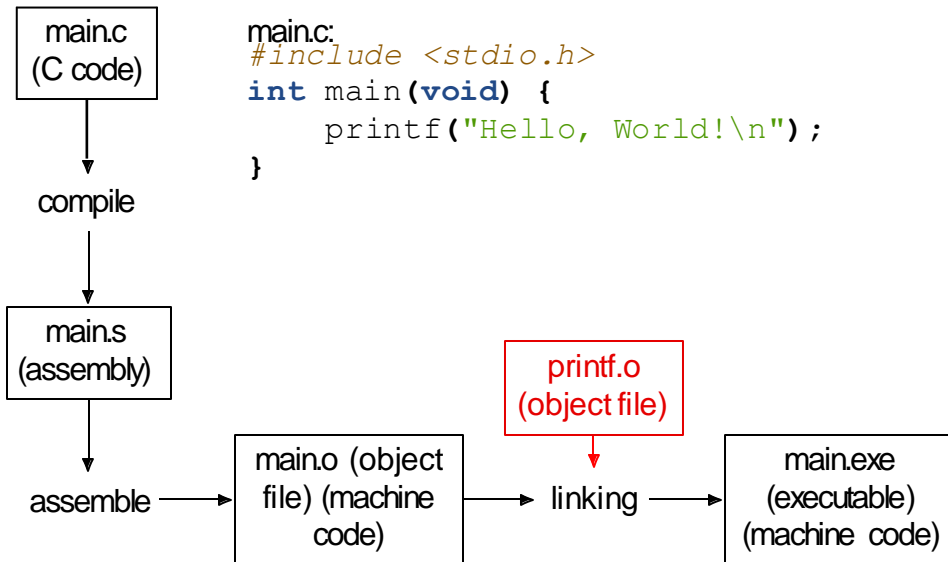


To write efficient code we also need to understand the process of going from c to machine code?

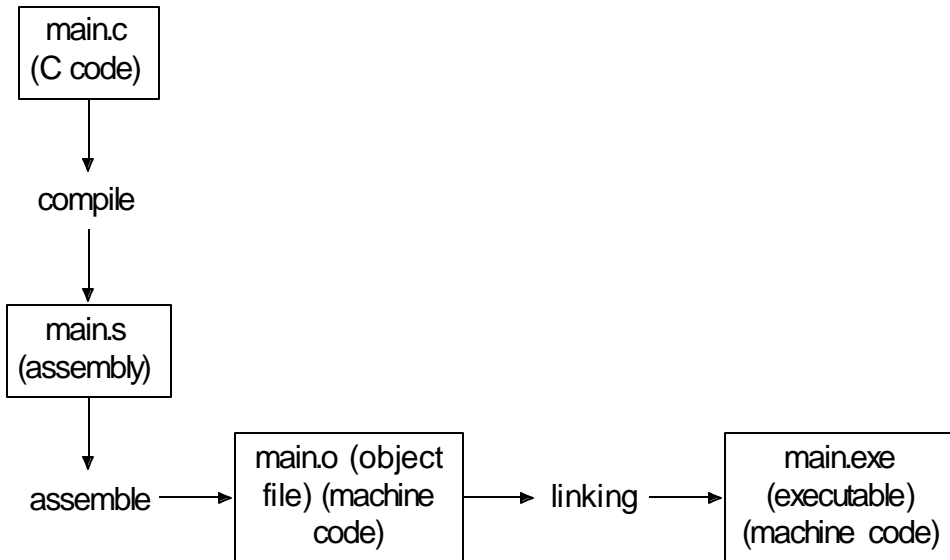
What does the compiler Do?



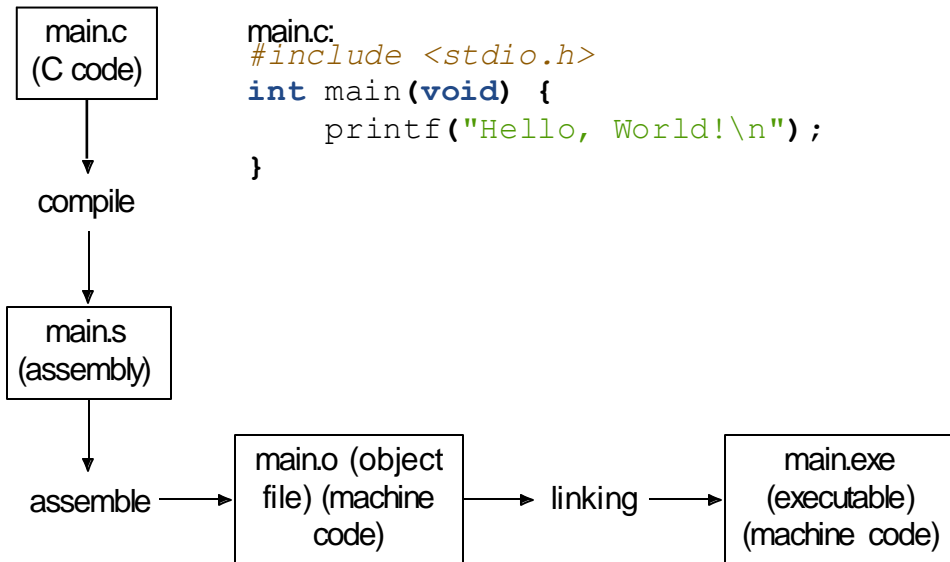
compilation pipeline



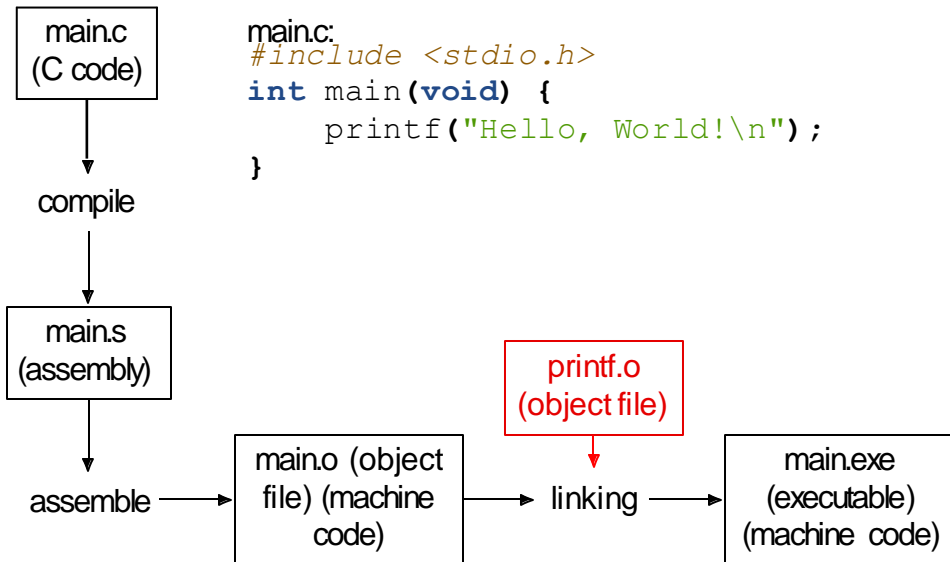
compilation pipeline



compilation pipeline



compilation pipeline



what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp
    mov $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret

.data
.Lstr: .string "Hello, World!"
```

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp
    mov $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret

.data
.Lstr: .string "Hello, World!"
```

101

101

000

Calling convention

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp
    mov $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret

.data
.Lstr: .string "Hello, World!"
```

hello.o

```
text (code) segment:
48 83 EC 08 BF 00 00 00 00 E8 00 00
00 00 31 C0 48 83 C4 08 C3
data segment:
48 65 6C 6C 6F 2C 20 57 6F 72 6C 00
relocations
    take 0s at          and replace with
    text, byte 6()     data segment, byte 0
    text, byte 10()    address of puts
symbol table:
main    text byte 0
```

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp
    mov $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret

.data
.Lstr: .string "Hello,_World!"
```

hello.o

text (code) segment:

```
48 83 EC 08 BF 00 00 00 00 E8 00 00
00 00 31 C0 48 83 C4 08 C3
```

data segment:

```
48 65 6C 6C 6F 2C 20 57 6F 72 6C 00
```

relocations:

take 0s at	and replace with
text, byte 6()	data segment, byte 0
text, byte 10()	address of puts

symbol table

main text byte 0

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp
    mov $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret

.data
.Lstr: .string "Hello,_World!"
```

hello.o

text (code) segment:

```
48 83 EC 08 BF 00 00 00 00 E8 00 00
00 00 31 C0 48 83 C4 08 C3
```

data segment:

```
48 65 6C 6C 6F 2C 20 57 6F 72 6C 00
```

relocations:

take 0s at
text, byte 6() and replace with
text, byte 10() address of puts
data segment, byte 0

symbol table

```
main text byte 0
```

Don't know
where it will
get put in
memory the
linker will fill
it in

Code puts

Code hello

Data puts

Data hello

Instructions for link to find where
put the addresses of data

what's in those files?

hello.c

```
#include <stdio.h>
int main(void) {
    puts("Hello, World!");
    return 0;
}
```

hello.s

```
.text
main:
    sub $8, %rsp    mov
    $.Lstr, %rdi
    call puts
    xor %eax, %eax
    add $8, %rsp
    ret
```

hello.o

text (code) segment:

```
48 83 EC 08 BF 00 00 00 00 E8 00 00
00 00 31 C0 48 83 C4 08 C3
```

data segment:

```
48 65 6C 6C 6F 2C 20 57 6F 72 6C 00
```

relocations:

take 0s at and replace with
text, byte 6() data segment, byte 0
text, byte 10() address of puts

symbol table

```
main    text byte 0
```

```
.data
.Lstr: .string "Hello, World!"
```

hello.exe

(actually binary, but shown as hexadecimal) ...

```
48 83 EC 08 BF A7 02 04 00
E8 08 4A 04 00 31 C0 48
83 C4 08 C3 ...
```

...(code from stdio.o) ...

```
48 65 6C 6C 6F 2C 20 57 6F
72 6C 00 ...
```

...(data from stdio.o) ...

Updated
with address
of hello
Word

+ stdio.o

Memory Address bus
Get me value at
x0003

Memory Data bus
Your data was:
xffff

Functional Block Diagram, MSP430G2x53

