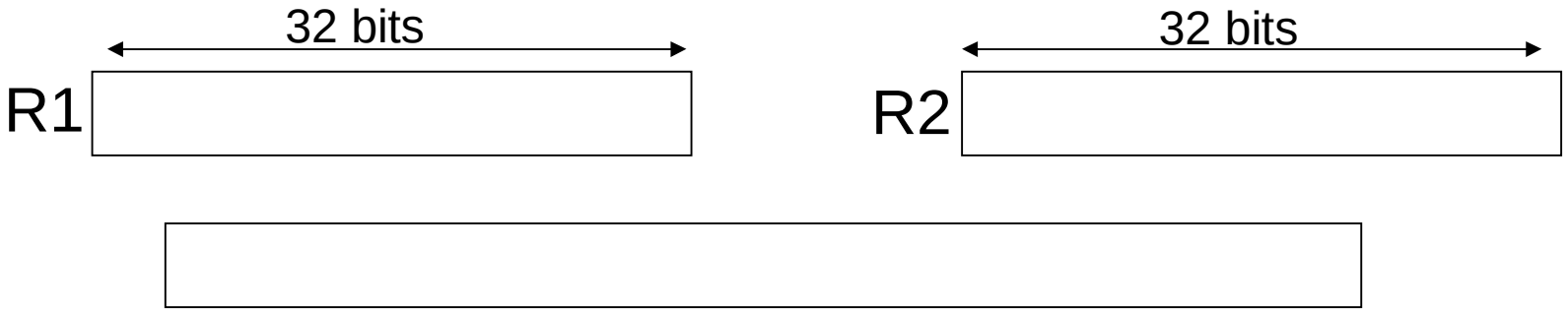
MULT Instruction

- Example: MULT R1, R2



The product could be twice as large (i.e., 64b)

It may not fit into a single general purpose register

- MIPS 1 Solution: Put the 32 least significant bits of the product in LO and the rest in HI
- For DIV: quotient in LO, remainder in HI

Control Transfer Instructions

	Mnemonics	Example	Meaning
Conditional Branch	BEQ, BNE, BGEZ, BLEZ, BLTZ, BGTZ	BLTZ R2, -16	If R2 < 0, PC ← PC + 4 -16
Jump	J, JR	J $target_{26}$	PC ← $(PC)_{31-28} \| \| target_{26} \| \| 00$
Jump and Link	JAL, JALR	JALR R2	R31 ← PC + 8 PC ← R2
System call	SYSCALL	SYSCALL	

R: Register

$target_{26}$: Absolute operand

Z: Zero

Control Transfer Instructions

	Mnemonics	Example	Meaning
Conditional Branch	BEQ, BNE, BGEZ, BLEZ, BLTZ, BGTZ	BLTZ R2, -16	If R2 < 0, PC ← PC + 4 -16
Jump	J, JR	J $target_{26}$	PC ← $(PC)_{31-28} \| \| target_{26} \| \| 00$
Jump and Link	JAL, JALR	JALR R2	R31 ← PC + 8 PC ← R2
System call	SYSCALL	SYSCALL	

Conditional Branch Instructions

	Mnemonics	Example	Meaning
Conditional Branch	BEQ, BNE, BGEZ, BLEZ, BLTZ, BGTZ	BLTZ R2, -16	If R2 < 0, PC ← PC + 4 -16

= = != >= 0 <= 0 < 0 > 0

Example: BEQ R1, R2, -16

- Take the branch if contents of R1 = = contents of R2
- The target address of the branch is specified using PC relative addressing mode
- PC + 4 -16
- For examples, we will use a more readable notation

  BEQ R1, R2, label

Jump Instructions

	Mnemonics	Example	Meaning
Jump	J, JR	J $target_{26}$	PC ← $(PC)_{31-28}$ || $target_{26}$ || 00
		JR R5	PC ← R5

- Unconditional control transfer to the instruction at the target address
  - Target address specified using absolute addressing mode
  - i.e. target address itself included in the jump instruction
  - But addresses are also 32b in size: an address won't fit into a 32b instruction
  - MIPS 1 Solution: Include 26bits of target address in the instruction

Jump and Link Instructions

	Mnemonics	Example	Meaning
Jump and Link	JAL, JALR	JALR R2 JAL $target_{26}$	R31 ← PC + 8 PC ← R2

- Unconditional control transfer to the instruction at the target address along with remembering of PC+8 in R31
  - Target address specified as in J, JR instructions
  - We will see how this instruction is useful in function calls

C Control Transfer Constructs…

C	MIPS 1 Instructions
goto label	J label
if (X > 0) thenpart; else elsepart	LW R1, X BGTZ R1, thenpart elsepart: : thenpart:
repeat loopbody until (X != 0)	LW R1, X loophead: loopbody BEQ R1, R0, loophead

Interesting Notes from ISA Manual.

- For load instructions: the loaded value might not be available in the destination register for use by the instruction immediately following the load
  - LOAD DELAY SLOT
- For control transfer instructions: the transfer of control takes place only following the instruction immediately after the control transfer instruction
  - BRANCH DELAY SLOT

Interesting Notes from ISA Manual.

- Warning about Load instructions

```
LW     R1,   -8(R2)
ADD    R3,  R1,  R2
```

Interesting Notes from ISA Manual

- For load instructions: the loaded value might not be available in the destination register for use by the instruction immediately following the load
  - LOAD DELAY SLOT
- For control transfer instructions: the transfer of control takes place only following the instruction immediately after the control transfer instruction
  - BRANCH DELAY SLOT

Interesting Notes from ISA Manual.

- Warning about Load instructions

```
LW    R1,  -8(R2)
ADD   R3,  R1,  R2
```

```
LW    R1,  -8(R2)
An instruction that does not use R1
ADD   R3,  R1,  R2
```

- Warning about Control Transfer Instructions

```
head:  loopbody
       BEQ  R1,  R0,  head
```

Interesting Notes from ISA Manual

- For load instructions: the loaded value might not be available in the destination register for use by the instruction immediately following the load
  - LOAD DELAY SLOT
- For control transfer instructions: the transfer of control takes place only following the instruction immediately after the control transfer instruction
  - BRANCH DELAY SLOT

Interesting Notes from ISA Manual.

- Warning about Load instructions

```
LW    R1,  -8(R2)
ADD   R3,  R1,  R2
```

```
LW    R1,  -8(R2)
An instruction that does not use R1
ADD   R3,  R1,  R2
```

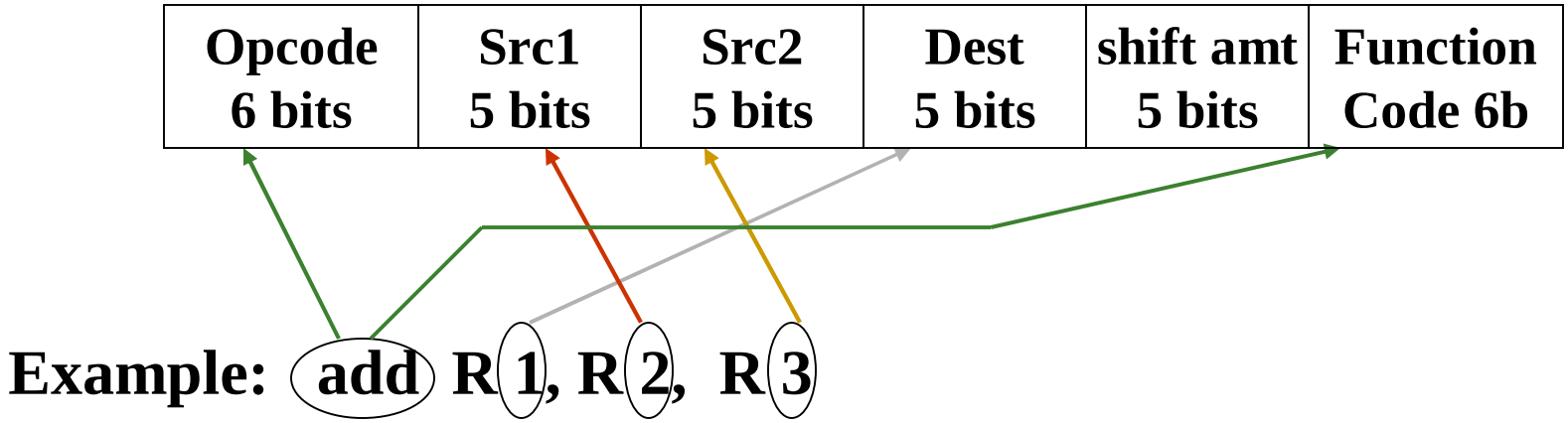
- Warning about Control Transfer Instructions

```
head:  loopbody
       BEQ  R1,  R0,  head
```

```
head:  loopbody
       BEQ  R1,  R0,  head
       next-instruction
```

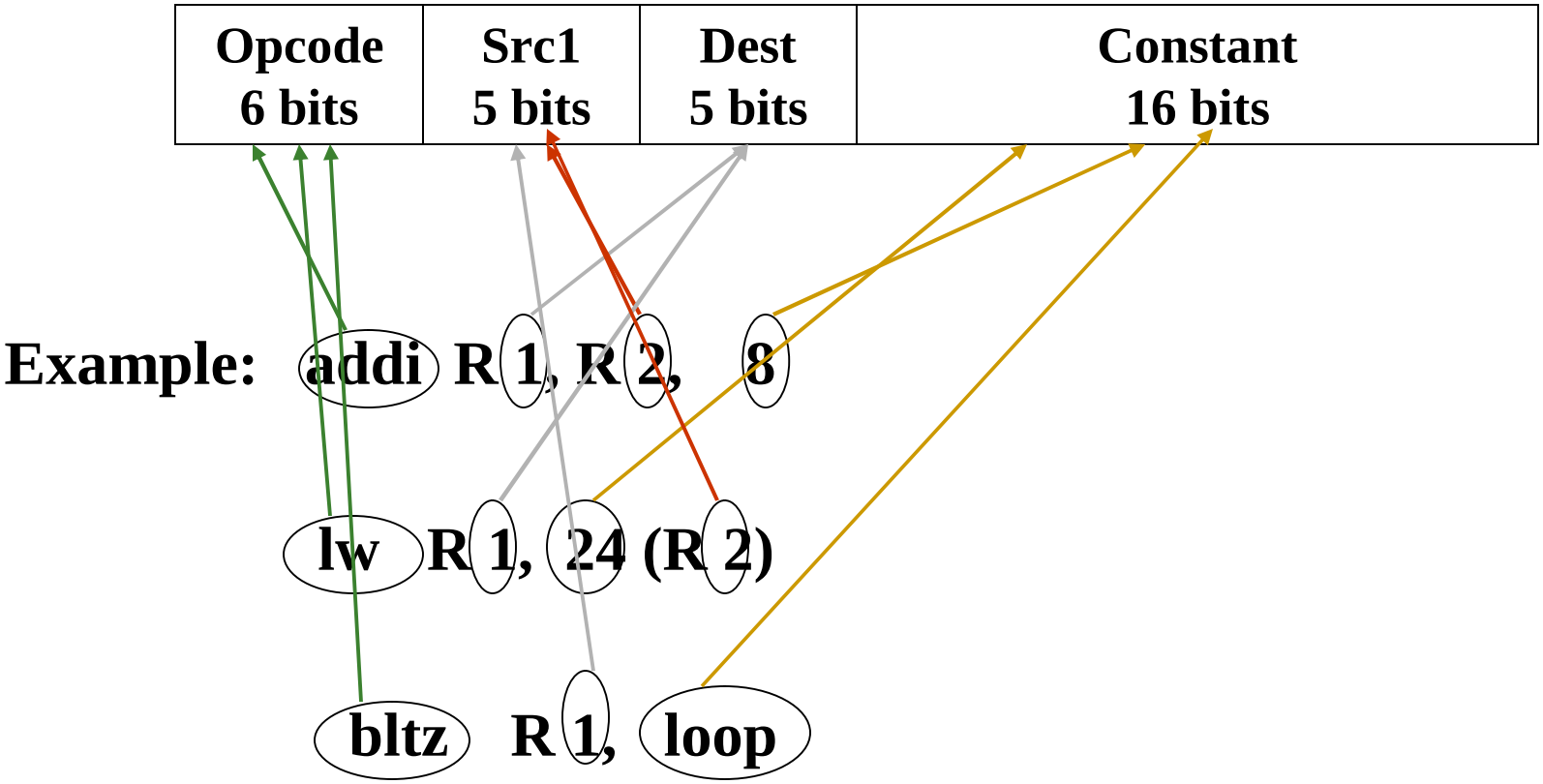
MIPS 1 Instruction Encoding

R Format



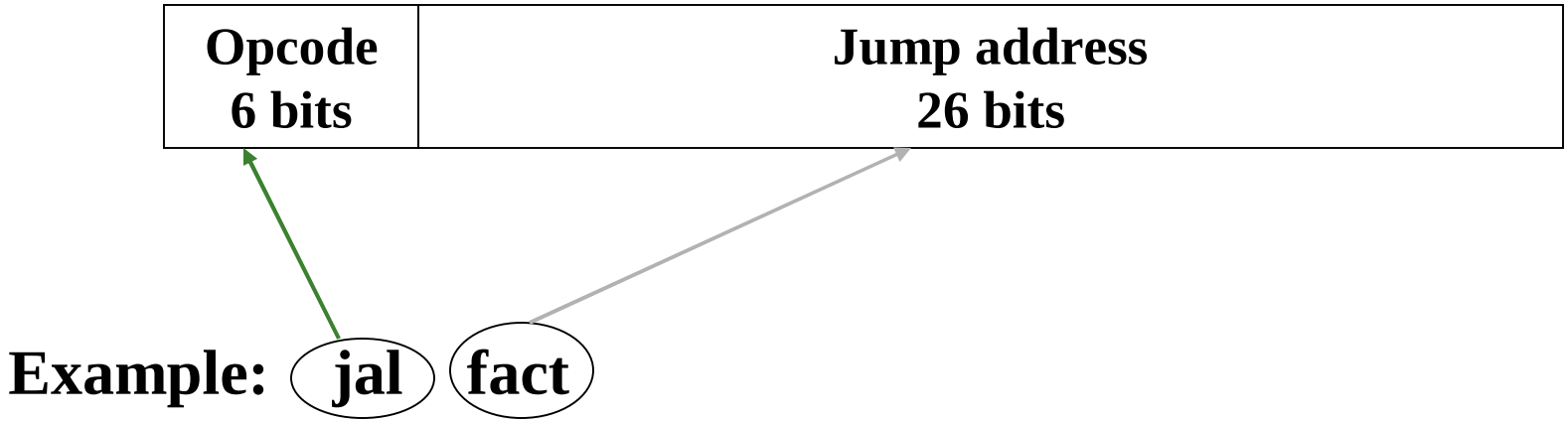
MIPS 1 Instruction Encoding

I Format



MIPS 1 Instruction Encoding

J Format

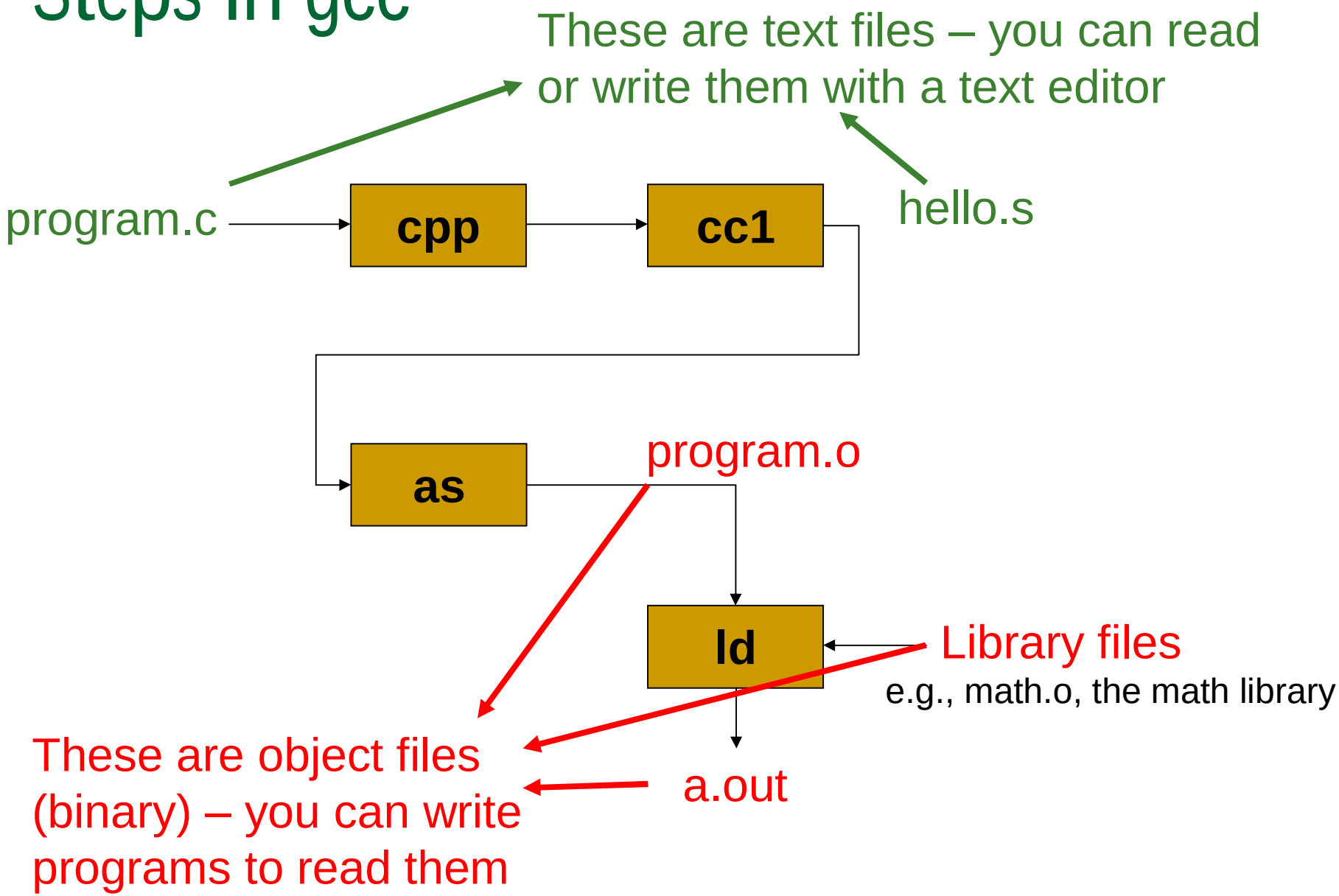


Recall: C Program to a.out

% gcc program.c

- program.c: File containing program written in the C programming language
- a.out: File containing executable equivalent program in machine language

Steps in gcc


These are text files – you can read or write them with a text editor
program.c
cpp
cc1
hello.s
as
program.o
ld
Library files
e.g., math.o, the math library
These are object files (binary) – you can write programs to read them
a.out