CS61C Guerrilla Section 2: MIPS and CALL

Led by Peijie, Connor and Sandy

June 30, 2016

Topics for Today

Topics for Today

- 1. MIPS Review
- 2. CALL

MIPS Review - Registers

- 32 registers, 32 bits wide, holds a word

MIPS Review - Registers

- 32 registers, 32 bits wide, holds a word
- Operands used by instructions

MIPS Review - Registers

- 32 registers, 32 bits wide, holds a word
- Operands used by instructions
- No data types, just raw bits.

MIPS Review - Registers

- 32 registers, 32 bits wide, holds a word
- Operands used by instructions
- No data types, just raw bits.
- EXTREMELY FAST

MIPS Review - Data Transfer

memop reg, off(bAddr)

MIPS Review - Data Transfer

memop reg, off(bAddr)

- memop = operation name ("operator")
- reg = register for operation source or destination
- bAddr = register with pointer to memory ("base address")
- off = address offset (immediate) in bytes ("offset")

MIPS Review - Data Transfer

memop reg, off(bAddr)

- memop = operation name ("operator")
- reg = register for operation source or destination
- bAddr = register with pointer to memory ("base address")
- off = address offset (immediate) in bytes ("offset")
- Accesses memory address bAddress + offset

MIPS Review - Data Transfer

memop reg, off(bAddr)

- memop = operation name ("operator")
- reg = register for operation source or destination
- bAddr = register with pointer to memory ("base address")
- off = address offset (immediate) in bytes ("offset")
- Accesses memory address bAddress + offset
- lw $t0, 12($s3)
- sw $t1, 40($a4)

MIPS Review - Instructions

- Arithmetic: add, sub, addi, muly, addu, subu, addiu
- Data Transfer: lw, sw, lb, sb
- Branching: beq, bne, slt, slti
- Bitwise: and, andi, or, ori, nor, xor, xori
- Shifting: sll, srl, sra
- Pesudo-Instruction: move, la, li

MIPS Review - Function Calls

MIPS Review - Function Calls

1. Put arguments in arguments registers $a*
2. Transfer control to the function: jal
3. The function acquires any storage resources it needs.
4. The function performs its desired task
5. The function returns and "cleans up": $v*
6. Control is returned to you: jr $ra

MIPS Review - Calling Convensions

Who needs to store their registers in the stack? And where to save them?

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	No

MIPS Review - Calling Convensions

Who needs to store their registers in the stack? And where to save them?

- $t*, $v*, $a* ¡— Caller (the calling function)
- $s*, $sp, $ra ¡— Callee (the function being called)

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	No

MIPS Review - Calling Convensions

Who needs to store their registers in the stack? And where to save them?

- $t*, $v*, $a* ¡— Caller (the calling function)
- $s*, $sp, $ra ¡— Callee (the function being called)
- $sp: pointer to last used space of stack.

REGISTER NAME, NUMBER, USE, CALL CONVENTION

NAME	NUMBER	USE	PRESERVED ACROSS A CALL?
$zero	0	The Constant Value 0	N.A.
$at	1	Assembler Temporary	No
$v0-$v1	2-3	Values for Function Results and Expression Evaluation	No
$a0-$a3	4-7	Arguments	No
$t0-$t7	8-15	Temporaries	No
$s0-$s7	16-23	Saved Temporaries	Yes
$t8-$t9	24-25	Temporaries	No
$k0-$k1	26-27	Reserved for OS Kernel	No
$gp	28	Global Pointer	Yes
$sp	29	Stack Pointer	Yes
$fp	30	Frame Pointer	Yes
$ra	31	Return Address	No

MIPS Review - Collecting the pieces

Basic Structure of a Function

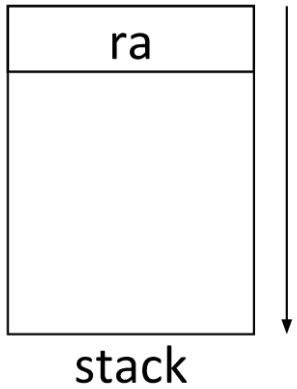
Prologue

```
func_label:
addi $sp,$sp, -framesize
sw $ra, <framesize-4>($sp)
save other regs if need be
```

Body (call other functions...)

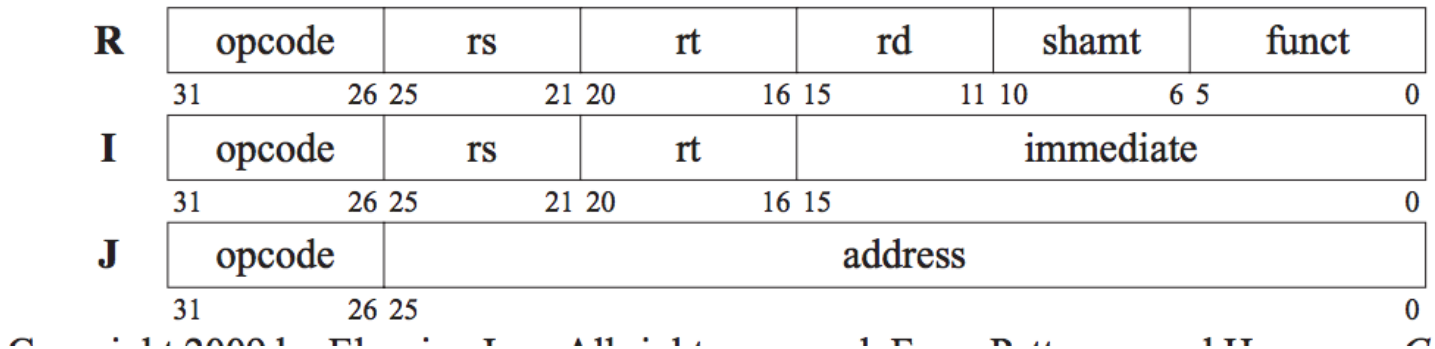
Epilogue

```
restore other regs if need be
lw $ra, <framesize-4>($sp)
addi $sp,$sp, framesize
jr $ra
```



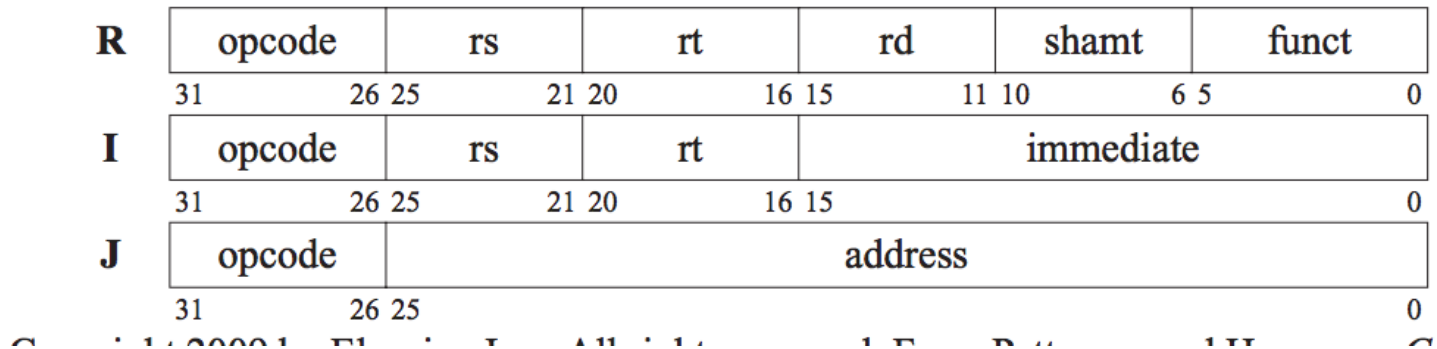
MIPS Review - Instruction Formats

BASIC INSTRUCTION FORMATS



MIPS Review - Instruction Formats

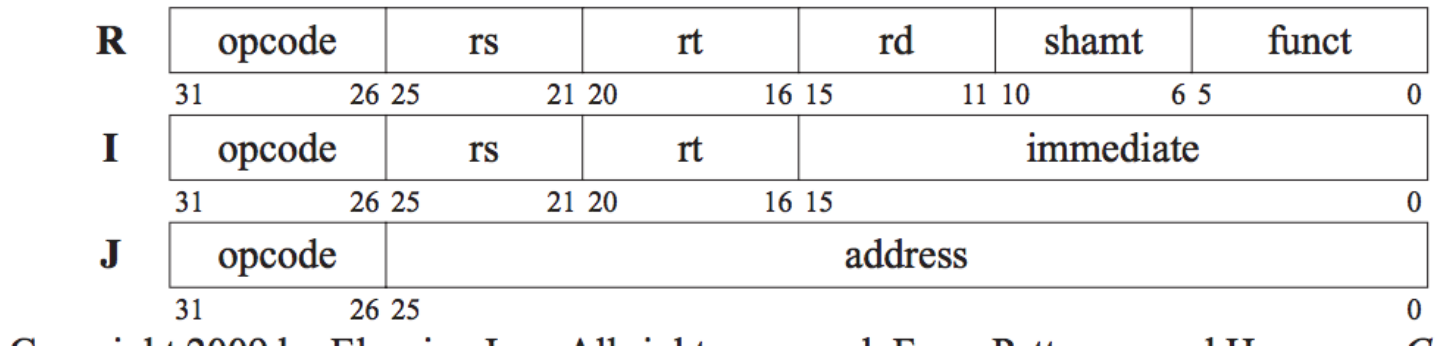
BASIC INSTRUCTION FORMATS



- I-Format: lw/sw, beq/bne, instr. with imm.
- J-Format: j and jal
- R-Format: all other instructions (including jr)

MIPS Review - Instruction Formats

BASIC INSTRUCTION FORMATS



- I-Format: lw/sw, beq/bne, instr. with imm.
- J-Format: j and jal
- R-Format: all other instructions (including jr)
- e.g. add rd, rs, rt

MIPS Review - Jumps and Branches

MIPS Review - Jumps and Branches

- J: Pseudo-direct Addressing:
- ——> PC = {PC+4}(31:28) + target address << 2

MIPS Review - Jumps and Branches

- J: Pseudo-direct Addressing:
- ——> PC = {PC+4}(31:28) + target address << 2
- Jr: Register Addressing
- ——> PC = full 32 bit address stored in src register

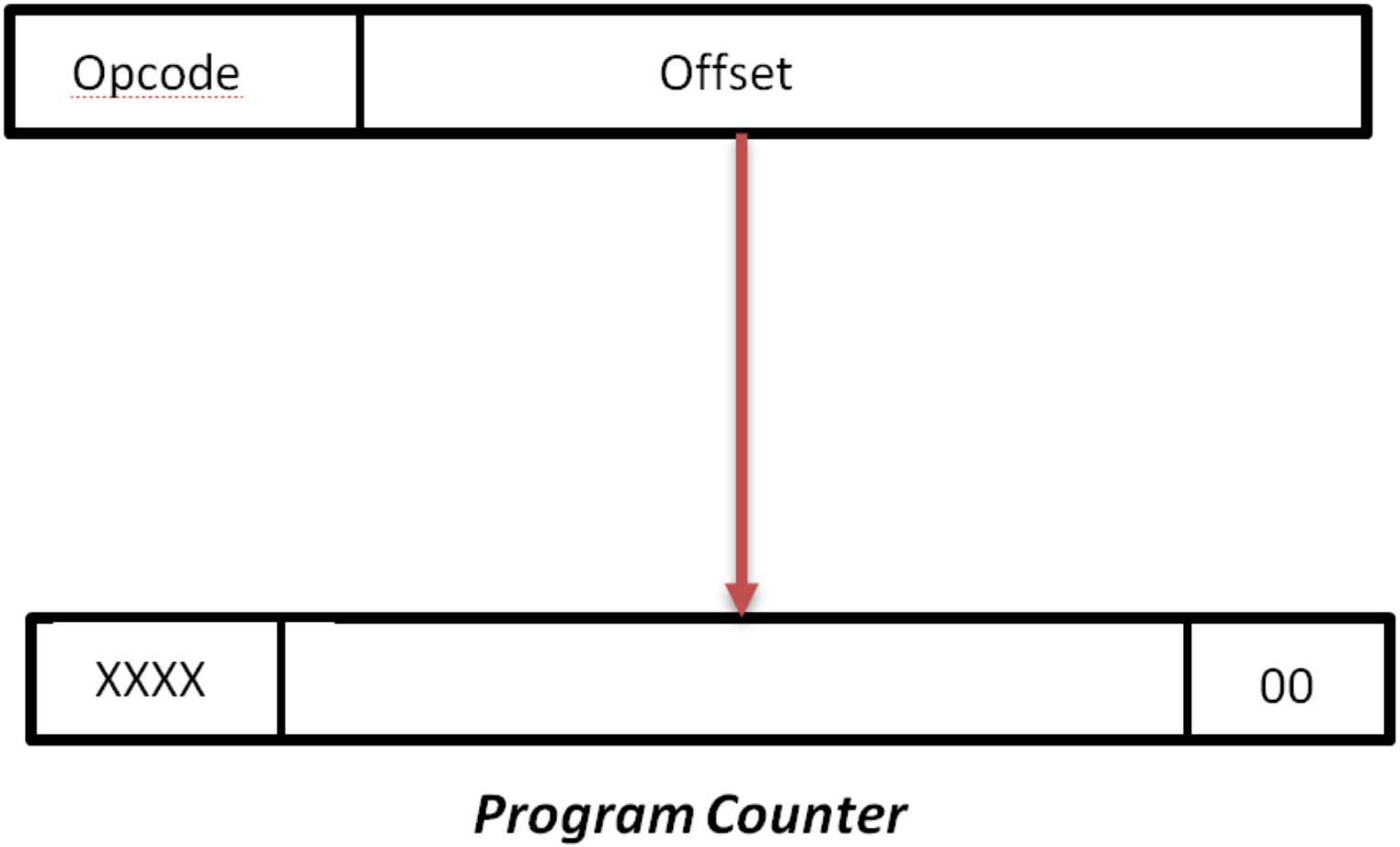
MIPS Review - Jumps and Branches

- J: Pseudo-direct Addressing:
- ——> PC = {PC+4}(31:28) + target address << 2
- Jr: Register Addressing
- ——> PC = full 32 bit address stored in src register
- beq/bne: PC-Relative Addressing
- ——> PC = PC +4+ imm << 2

MIPS Review - Jumps and Branches

- J: Pseudo-direct Addressing:
- ——> PC = {PC+4}(31:28) + target address << 2
- Jr: Register Addressing
- ——> PC = full 32 bit address stored in src register
- beq/bne: PC-Relative Addressing
- ——> PC = PC +4+ imm << 2
- lw/lb/sw/sb: Base Addressing
- ——> Mem[register] + immediate

MIPS Review - Pseudo-direct Addressing



MIPS Review - Jump/Branch: Range?

MIPS Review - Jump/Branch: Range?

- J: Pseudo-direct Addressing:
- ——> 2^{26} instructions;
- ——> maximum $2^{26} << 2 = 2^{28}$ bytes away

MIPS Review - Jump/Branch: Range?

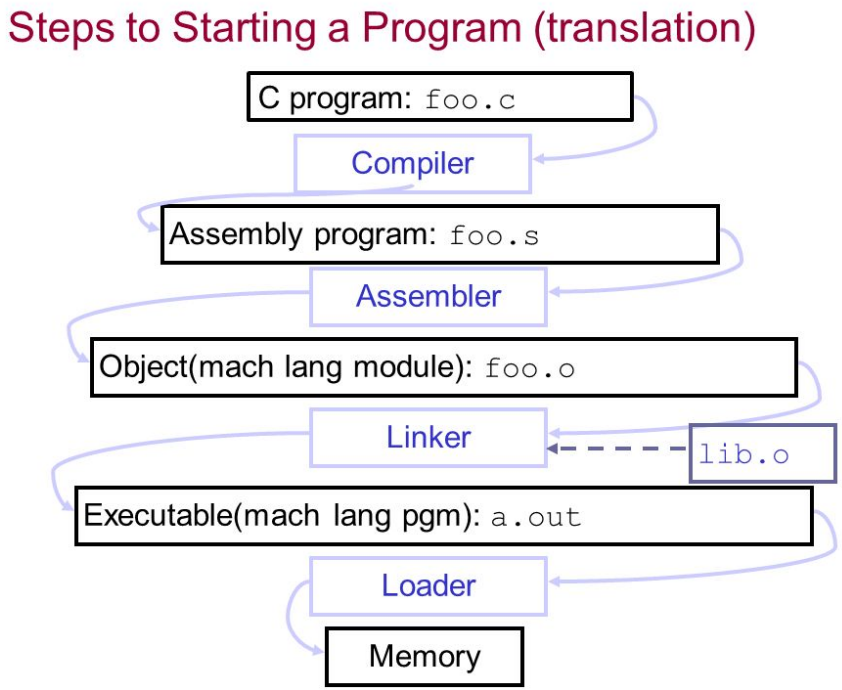
- J: Pseudo-direct Addressing:
- ——> 2^{26} instructions;
- ——> maximum $2^{26} << 2 = 2^{28}$ bytes away
- Jr: Register Addressing
- ——> 2^{32} instructions;

MIPS Review - Jump/Branch: Range?

- J: Pseudo-direct Addressing:
- ——> 2^{26} instructions;
- ——> maximum $2^{26} << 2 = 2^{28}$ bytes away
- Jr: Register Addressing
- ——> 2^{32} instructions;
- beq/bne: PC-Relative Addressing
- ——> 2^{16} instructions;
- ——> maximum 2^{17} bytes away in either direction

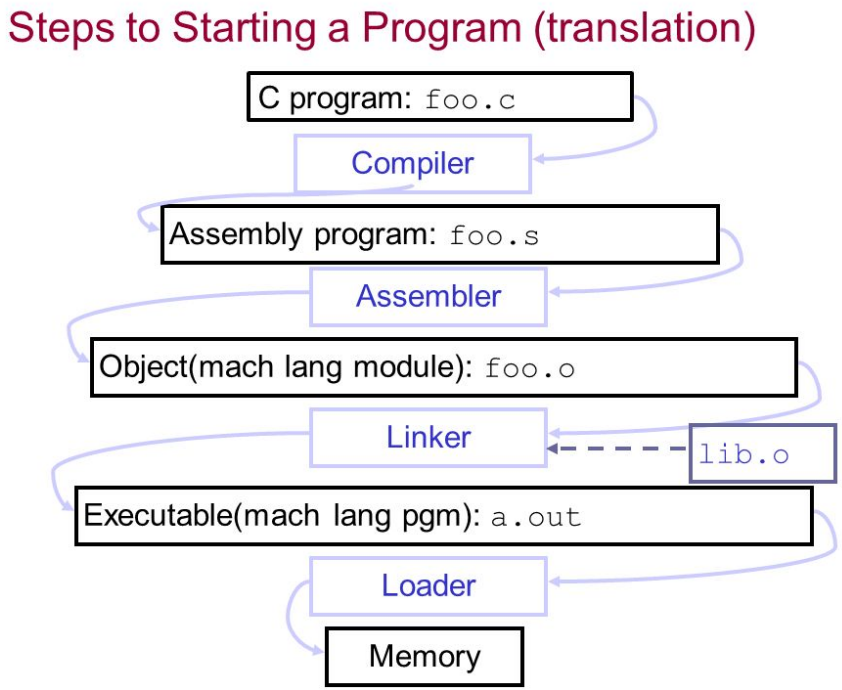
CALL - Translation

- C: produces MAL code



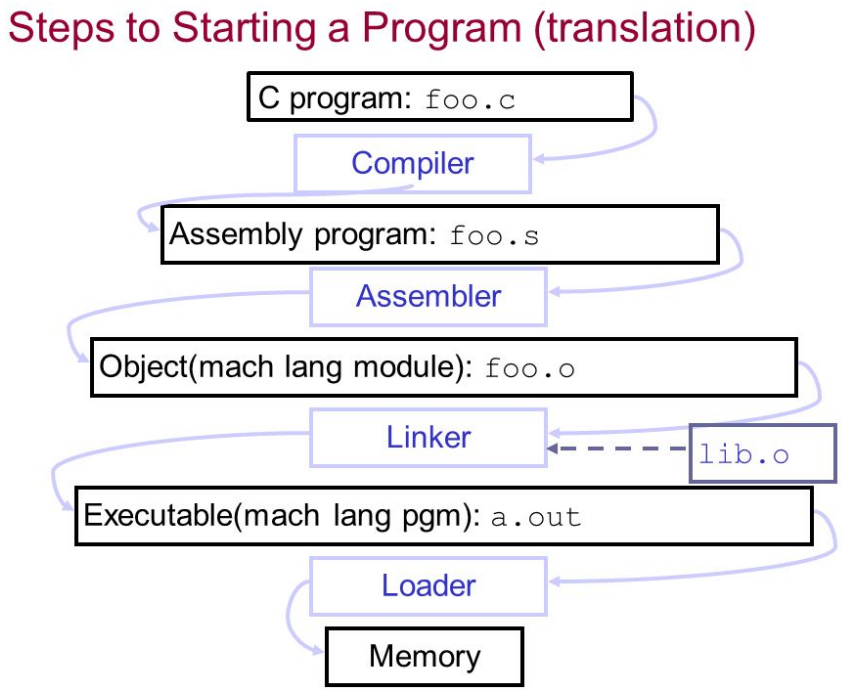
CALL - Translation

- C: produces MAL code
- A: 2 Passes, removes pseudo-instructions, produce TAL, Symbol Table and Relocation Table



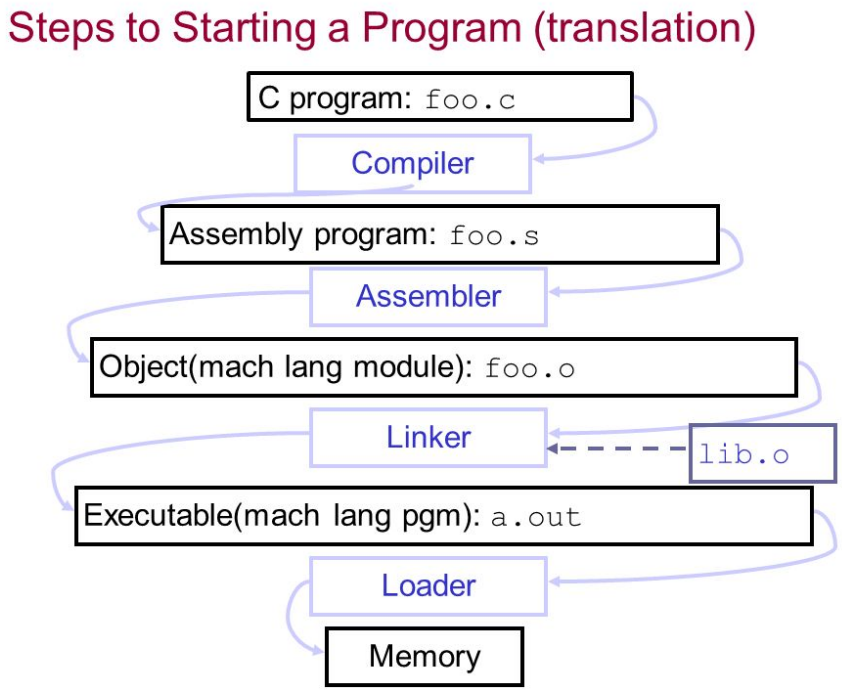
CALL - Translation

- C: produces MAL code
- A: 2 Passes, removes pseudo-instructions, produce TAL, Symbol Table and Relocation Table
- L: combines object files and resolve absolute addresses



CALL - Translation

- C: produces MAL code
- A: 2 Passes, removes pseudo-instructions, produce TAL, Symbol Table and Relocation Table
- L: combines object files and resolve absolute addresses
- L: loads executable and run program



WELCOME TO THE WORLD OF MIPS!