TED ÜNİVERSİTESİ

CMPE361 Computer Organization

Department of Computer Engineering
TED University- Fall 2023

Compiling, Assembling, Linking, Loading

These Slides are mainly based on slides of the text book (downloadable from the book’s website).

Example: From programming to execution

Program: C Code

```
# main calls function sum() to compute sum of arguments a and b
int f, g, y;  // global variables
int main(void)
{
  f = 2;
  g = 3;
  y = sum(f, g);
  return y;
}

int sum(int a, int b) {
  return (a + b);
}
```

Example Program: MIPS Assembly, after compiling

```
int f, g, y;  // global


int main(void)
{


  f = 2;
  g = 3;

  y = sum(f, g);
  return y;
}

int sum(int a, int b) {
  return (a + b);
}
```

```
.data #directive for assembler
f:
g:
y:
.text #directive for assembler
main: # Assebler assumes $gp set to 0x10008000, middle of
    global data area…
  addi $sp, $sp, -4    # stack frame
  sw   $ra, 0($sp)     # store $ra
  addi $a0, $0, 2      # $a0 = 2
  sw   $a0, f          # f = 2        #sw $a0, 0x8000 ($gp)
  addi $a1, $0, 3      # $a1 = 3
  sw   $a1, g          # g = 3        #sw $a1, 0x8004 ($gp)
  jal  sum             # call sum
  sw   $v0, y          # y = sum()    #sw $v0, 0x8008 ($gp)
  lw   $ra, 0($sp)     # restore $ra
  addi $sp, $sp, 4     # restore $sp
  jr   $ra             # return to OS
sum:
  add  $v0, $a0, $a1   # $v0 = a + b
  jr   $ra             # return
```

Symbol Table maintained by the assembler

Symbol	Address

- Symbol Table is created during the first phase of assembler.
- Symbol addresses are based on where they are placed in the **static data** or **heap** area.
- Instruction addresses are based on where they are placed in the **text** area

Symbol Table, after the first pass

Symbol	Address
`f`	**0x10000000**
`g`	**0x10000004**
`y`	**0x10000008**
`main`	**0x00400000**
`sum`	**0x0040002C**

Heap starts at 0x10000000
Text starts at 0x40000000

Executable file

Executable file header	Text Size	Data Size
	0x34 (52 bytes)	0xC (12 bytes)
Text segment	**Address**	**Instruction**
	0x00400000	0x23BDFFFC
	0x00400004	0xAFBF0000
	0x00400008	0x20040002
	0x0040000C	0xAF848000
	0x00400010	0x20050003
	0x00400014	0xAF858004
	0x00400018	0x0C10000B
	0x0040001C	0xAF828008
	0x00400020	0x8FBF0000
	0x00400024	0x23BD0004
	0x00400028	0x03E00008
	0x0040002C	0x00851020
	0x00400030	0x03E00008
Data segment	**Address**	**Data**
	0x10000000	f
	0x10000004	g
	0x10000008	y

```
addi $sp, $sp, -4
sw   $ra, 0 ($sp)
addi $a0, $0, 2
sw   $a0, 0x8000 ($gp)
addi $a1, $0, 3
sw   $a1, 0x8004 ($gp)
jal  0x0040002C
sw   $v0, 0x8008 ($gp)
lw   $ra, 0 ($sp)
addi $sp, $sp, -4
jr   $ra
add  $v0, $a0, $a1
jr   $ra
```

Correction:
addi $sp,$sp, 4

Example Program: Executable

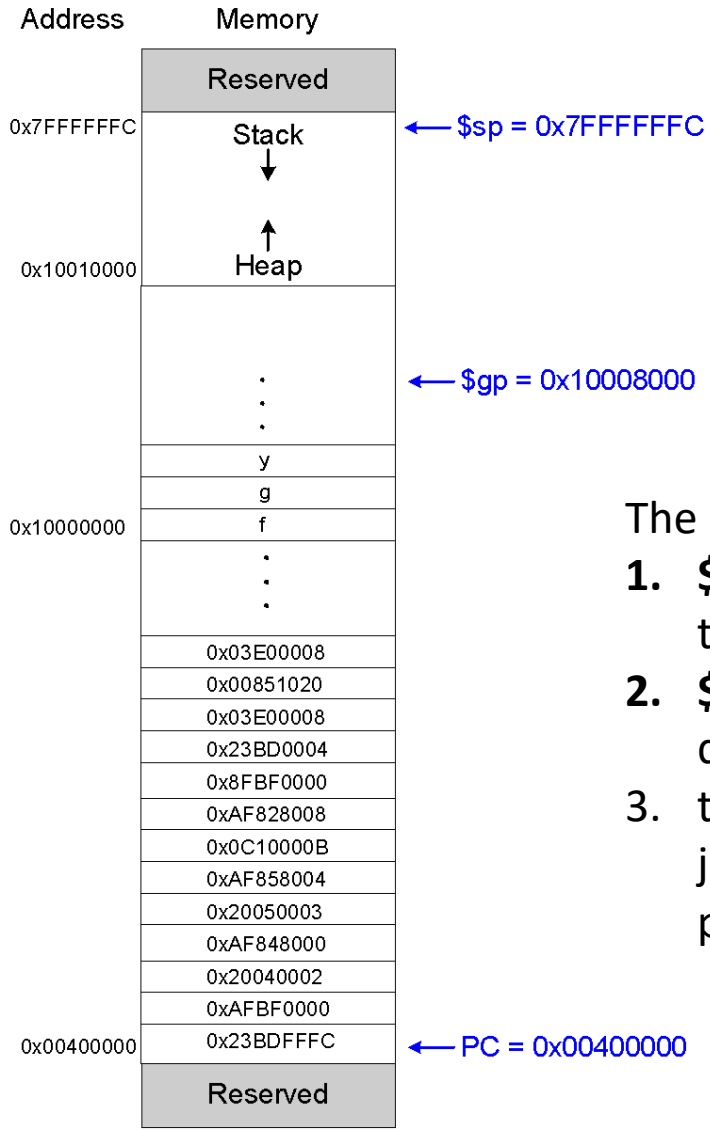
Executable file header	Text Size	Data Size
	0x34 (52 bytes)	0xC (12 bytes)
Text segment	**Address**	**Instruction**
	0x00400000	0x23BDFFFC
	0x00400004	0xAFBF0000
	0x00400008	0x20040002
	0x0040000C	0xAF848000
	0x00400010	0x20050003
	0x00400014	0xAF858004
	0x00400018	0x0C10000B
	0x0040001C	0xAF828008
	0x00400020	0x8FBF0000
	0x00400024	0x23BD0004
	0x00400028	0x03E00008
	0x0040002C	0x00851020
	0x00400030	0x03E00008
Data segment	**Address**	**Data**
	0x10000000	f
	0x10000004	g
	0x10000008	y

```
addi $sp, $sp, -4
sw   $ra, 0 ($sp)
addi $a0, $0, 2
sw   $a0, 0x8000 ($gp)
addi $a1, $0, 3
sw   $a1, 0x8004 ($gp)
jal  0x0040002C
sw   $v0, 0x8008 ($gp)
lw   $ra, 0 ($sp)
addi $sp, $sp, -4
jr   $ra
add  $v0, $a0, $a1
jr   $ra
```

Correction:
addi $sp,$sp, 4

StaticDataAddress for f=
$gp+signExtend(0x8000)
=0x10008000 + 0xFFFF8000
= 0x10000000

Executable in Memory



$gp: Global pointer,
Set by the loader or OS

The operating system sets

1. **$gp** to 0x10008000 (the middle of the global data segment)
2. **$sp** to 0x7FFFFFFC (the top of the dynamic data segment)
3. then performs a **jal** 0x00400000 to jump to the beginning of the program to be executed.