

CC4301

Arquitectura de computadores

Assembler Risc-V

3^{era} clase

Temario: registros resguardados de Risc-V, compilación optimizada, registro de activación, frame pointer, secciones del archivo assembler, especificación de Risc-V, codificación de instrucciones, punto flotante

Registros resguardados: *s0-s11*

Convención usada por los compiladores:

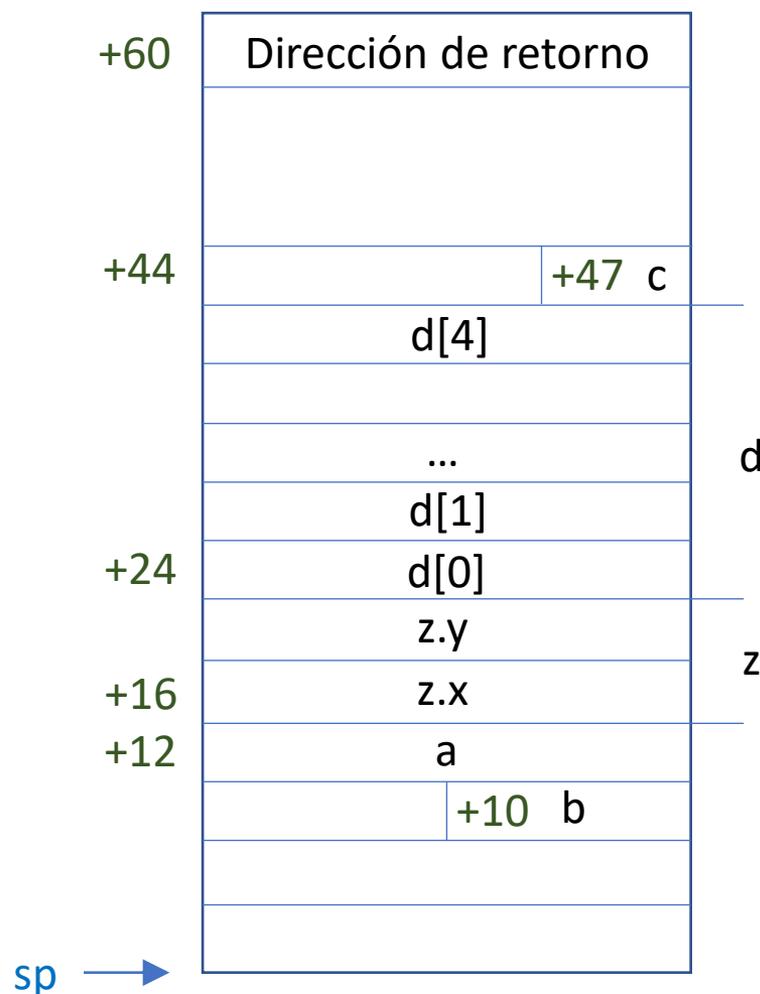
- Al programar una función puede usar *a0-a7* y *t0-t6* sin preocuparse de restaurarlos al retornar
- Puede modificar los registros *s0-s11*, pero debe restaurarlos al retornar
- Si llama a otra función perderá cualquier valor que haya dejado en *a0-a7* y *t0-t6*
- Si llama a otra función, se garantiza que cualquier valor que haya dejado en *s0-s11* sigue estando ahí
- Ejemplo: ver uso de registro *s0* en *g.c*

Compilación optimizada

- El optimizador mantiene en lo posible las variables en los registros del procesador
- Evalúa una sola vez expresiones comunes
- Evalúa funciones simples durante la compilación
- Ejemplo: ver assembler para *optim.c*

El registro de activación

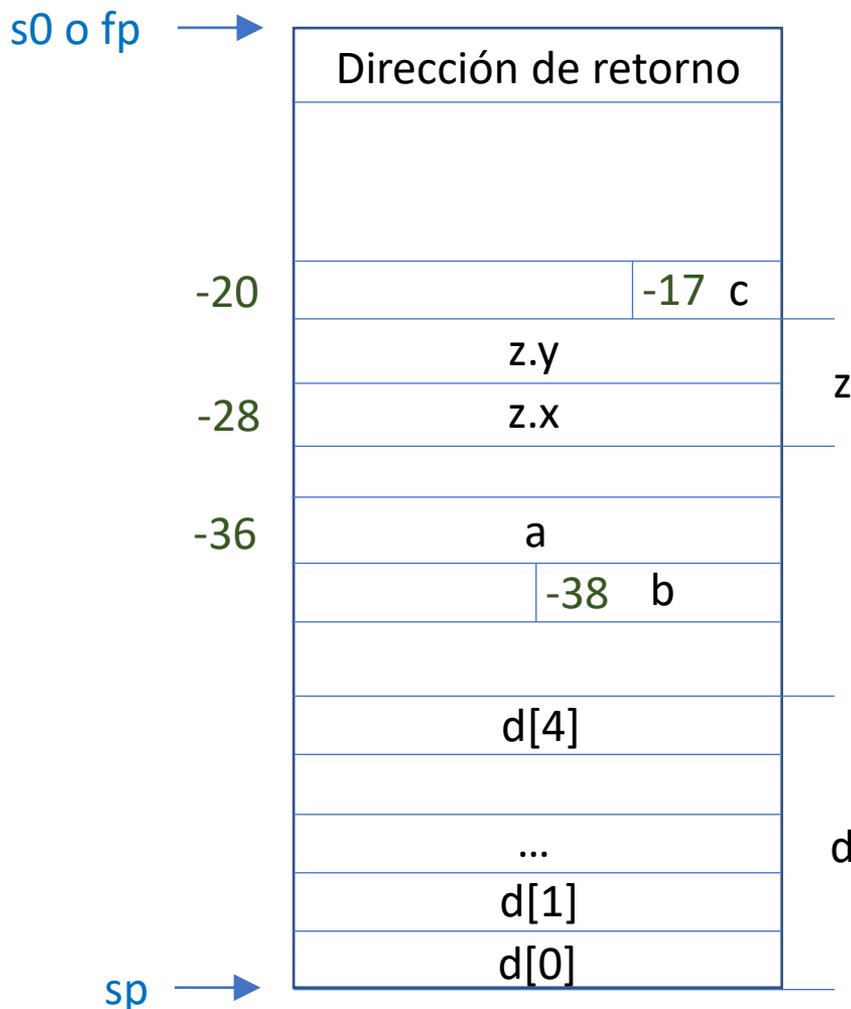
- Almacena las variables locales de una función
- Se apilan al ingresar a una función
- Se desapilan al retornar
- Ejemplo: ver assembler para *frame.c*



Arreglos y estructuras deben estar contiguos en memoria

Registro s0: frame pointer (fp)

- Se necesita cuando se declaran arreglos locales de tamaño desconocido en tiempo de compilación
- Apunta hacia el final del registro de activación
- Se usa para acceder a las variables locales en vez de sp. Ejemplo: ver assembler para *frame-pointer.c*



Los arreglos de tamaño desconocido en compilación quedan abajo en el registro de activación.

El resto de las variables locales están a una distancia de s0 conocida en tiempo de compilación.

Secciones de un archivo assembler

- Código: `.text`
- Datos inicializados: `.data` (o `.section .sdata`)
- Datos no inicializados: `.bss` (o `.section .sbss`)
- Datos de solo lectura: `.section .rodata`
- Ejemplo: ver assembler para *glob.c*
- Otras directivas:
 - Declarar etiquetas globales: `.globl`
 - Alinear a un tamaño potencia de 2^n : `.align n`
 - Declarar espacio para enteros (int): `.word`
 - Declarar espacio para enteros (short): `.half`
 - Declarar espacio para enteros (char): `.byte`
 - Declarar espacio para strings constantes: `.string`
 - Declarar n bytes para espacio en cero: `.zero n`
 - Declarar etiqueta de una función f: `.type f, @function`
 - % de una variable global g: `.type g, @object`

La especificación de Risc-V

- Sitio web: <https://riscv.org/>
- Especificaciones: <https://riscv.org/technical/specifications/>
- Instrucciones no privilegiadas: <https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf>
- Tutoriales:
 - googlear RiscV assembler tutorial
 - recomendar en el foro por favor

Codificación de instrucciones

RV32I Base Instruction Set

imm[31:12]				rd	0110111	LUI
imm[31:12]				rd	0010111	AUIPC
imm[20 10:1 11 19:12]				rd	1101111	JAL
imm[11:0]		rs1	000	rd	1100111	JALR
imm[12 10:5]	rs2	rs1	000	imm[4:1 11]	1100011	BEQ
imm[12 10:5]	rs2	rs1	001	imm[4:1 11]	1100011	BNE
imm[12 10:5]	rs2	rs1	100	imm[4:1 11]	1100011	BLT
imm[12 10:5]	rs2	rs1	101	imm[4:1 11]	1100011	BGE
imm[12 10:5]	rs2	rs1	110	imm[4:1 11]	1100011	BLTU
imm[12 10:5]	rs2	rs1	111	imm[4:1 11]	1100011	BGEU
imm[11:0]		rs1	000	rd	0000011	LB
imm[11:0]		rs1	001	rd	0000011	LH
imm[11:0]		rs1	010	rd	0000011	LW
imm[11:0]		rs1	100	rd	0000011	LBU
imm[11:0]		rs1	101	rd	0000011	LHU
imm[11:5]	rs2	rs1	000	imm[4:0]	0100011	SB
imm[11:5]	rs2	rs1	001	imm[4:0]	0100011	SH
imm[11:5]	rs2	rs1	010	imm[4:0]	0100011	SW
imm[11:0]		rs1	000	rd	0010011	ADDI
imm[11:0]		rs1	010	rd	0010011	SLTI
imm[11:0]		rs1	011	rd	0010011	SLTIU
imm[11:0]		rs1	100	rd	0010011	XORI
imm[11:0]		rs1	110	rd	0010011	ORI
imm[11:0]		rs1	111	rd	0010011	ANDI
0000000	shamt	rs1	001	rd	0010011	SLLI
0000000	shamt	rs1	101	rd	0010011	SRLI
0100000	shamt	rs1	101	rd	0010011	SRAI
0000000	rs2	rs1	000	rd	0110011	ADD
0100000	rs2	rs1	000	rd	0110011	SUB
0000000	rs2	rs1	001	rd	0110011	SLL
0000000	rs2	rs1	010	rd	0110011	SLT
0000000	rs2	rs1	011	rd	0110011	SLTU
0000000	rs2	rs1	100	rd	0110011	XOR
0000000	rs2	rs1	101	rd	0110011	SRL
0100000	rs2	rs1	101	rd	0110011	SRA
0000000	rs2	rs1	110	rd	0110011	OR
0000000	rs2	rs1	111	rd	0110011	AND

Punto flotante

- No es obligatorio implementarlo
- Extensión rv32f: operaciones con *float*
- Extensión rv32d: operaciones con *double*

RV32F Standard Extension

imm[11:0]		rs1	010	rd	0000111	FLW
imm[11:5]		rs2	rs1	010	imm[4:0]	FSW
rs3	00	rs2	rs1	rm	rd	FMADD.S
rs3	00	rs2	rs1	rm	rd	FMSUB.S
rs3	00	rs2	rs1	rm	rd	FNMSUB.S
rs3	00	rs2	rs1	rm	rd	FNMADD.S
0000000		rs2	rs1	rm	rd	FADD.S
0000100		rs2	rs1	rm	rd	FSUB.S
0001000		rs2	rs1	rm	rd	FMUL.S
0001100		rs2	rs1	rm	rd	FDIV.S
0101100		00000	rs1	rm	rd	FSQRT.S
0010000		rs2	rs1	000	rd	FSGNJ.S
0010000		rs2	rs1	001	rd	FSGNJN.S
0010000		rs2	rs1	010	rd	FSGNJX.S
0010100		rs2	rs1	000	rd	FMIN.S
0010100		rs2	rs1	001	rd	FMAX.S
1100000		00000	rs1	rm	rd	FCVT.W.S
1100000		00001	rs1	rm	rd	FCVT.WU.S
1110000		00000	rs1	000	rd	FMV.X.W
1010000		rs2	rs1	010	rd	FEQ.S
1010000		rs2	rs1	001	rd	FLT.S
1010000		rs2	rs1	000	rd	FLE.S
1110000		00000	rs1	001	rd	FCLASS.S
1101000		00000	rs1	rm	rd	FCVT.S.W
1101000		00001	rs1	rm	rd	FCVT.S.WU
1111000		00000	rs1	000	rd	FMV.W.X

Próxima clase

- La última arquitectura CISC: Intel/AMD x86
- Assembler Intel/Amd x86