Universidade de Brasília
Departamento de Ciência da Computação

Introdução aos Sistemas Computacionais

Disciplina: 113468

Prof. Marcus Vinicius Lamar

Universidade de Brasília
Departamento de Ciência da Computação

Assembly RISC-V Básico

Assembly RV32I

Exemplo: Estrutura de decisão if-then-else

Código de alto nível

```
if (i==j)
    f=g+h;
else
    f=f-i;
...
```

Código Assembly

```
# s0=f, s1=g, s2=h, s3=i, s4=j
      bne s3,s4,ELSE
      add s0,s1,s2
      jal x0,DONE
ELSE: sub s0,s0,s3
DONE:   ...
```

LABEL: Endereço de 32 bits
Alocação variável-registrador é feita pelo compilador

Assembly RV32I

Exemplo: Estrutura de decisão switch - case

Código de alto nível

```
switch(amount)
{
case 20: fee=5;
         break;

case 50: fee=3;
         break;

case 100: fee=2;
          break;

default: fee=0;
}
...
```

Código Assembly

```
# s0=amount, s1=fee
CASE20: addi t0, zero, 20
        bne s0, t0, CASE50
        addi s1, zero, 5
        jal x0, DONE
CASE50: addi t0, zero, 50
        bne s0, t0, CASE100
        addi s1, zero, 3
        jal x0, DONE
CASE100: addi t0, zero, 100
        bne s0, t0, DEFAULT
        addi s1, zero, 2
        jal x0, DONE
DEFAULT: addi s1, zero, 0
DONE: ...
```

Assembly RV32I

Exemplo: Estrutura de repetição while

Código de alto nível

```
int pow=1;
int x=0;

while(pow!=128)
{
  pow=pow*2;
  x=x+1;
}
...
```

Código Assembly

```
# s0=pow, s1=x
        addi s0,zero,1
        addi s1,zero,0
        addi t0,zero,128
WHILE: beq s0,t0,DONE
        slli s0,s0,1
        addi s1,s1,1
        jal x0,WHILE
DONE:   ...
```

Assembly RV32I

Exemplo: Estrutura de repetição for

Código de alto nível

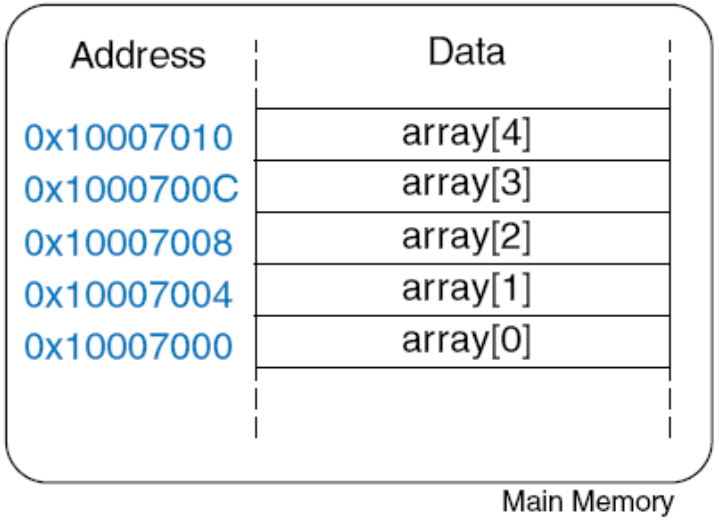
```
int sum=0;

for(i=0;i<10;i++)
{
  sum=sum+i;
}
...
```

Código Assembly

```
# s0=i, s1=sum
      add s1,zero,zero
      addi s0,zero,0
FOR:  slti t0,s0,10
      beq t0,zero,DONE
      add s1,s1,s0
      addi s0,s0,1
      jal x0, FOR
DONE: ...
```

Assembly RV32I

Exemplos de códigos utilizando arrays

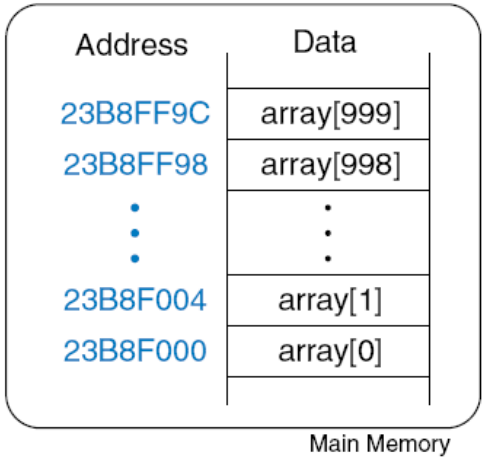



```
int array[5];

array[0]=array[0]*8;
array[1]=array[1]*8;
...
```

```
# s0= endereço base do array
      lui s0,0x10007
      lw t1,0(s0)
      slli t1,t1,3
      sw t1,0(s0)
      lw t1,4(s0)
      slli t1,t1,3
      sw t1,4(s0)
      ...
```

Assembly RV32I

Exemplos de códigos utilizando arrays




```
int i;
int array[1000];

for(i=0;i<1000;i++)
    array[i]=array[i]*8;

...
```

```
# s0= endereço base do array
# s1=i
        lui s0,0x23B8F
        add s1,zero,zero
        addi t2,zero,1000

LOOP:  bge s1,t2,DONE
        slli t0,s1,2
        add t0,s0,t0
        lw t1,0(t0)
        slli t1,t1,3
        sw t1,0(t0)
        addi s1,s1,1
        jal x0,LOOP
DONE:   ...
```

Assembly RV32I

Chamada de Procedimentos

Registrador ra: Endereço de Retorno

```
int main() {
   ...
   simple();
   ...
}

void simple() {
   return;
}
```

```
main: ...
      jal ra,simple
      ...

simple: jalr x0,ra,0
```

Convenção:

Registradores a0~a7	Argumentos das funções
Registradores a0~a1	Retorno dos valores

Assembly RV32I

Chamada de Procedimentos

```
int main() {
  int y;
  ...
  y=diffofsums(2,3,4,5);
  ...
}

int diffofsums(int f, int g,
 int h, int i)
{
  int result;

  result=(f+g)-(h+i);
  return result;
}
```

```
# s0= y
main:   ...
        addi a0,zero,2
        addi a1,zero,3
        addi a2,zero,4
        addi a3,zero,5
        jal ra,diffofsums
        add s0,a0,zero
        ...

diffofsums:
        add t0,a0,a1
        add t1,a2,a3
        sub a0,t0,t1
        jalr x0,ra,0
```

E se tiver mais argumentos ou mais valores de retorno? RAM

Assembly RV32IM**F** – Ponto Flutuante

- Aritmética usando padrão IEEE754
- 32 Registradores de precisão simples: `f0,f1,...,f30,f31`
- Operações com precisão simples.

Obs.: Precisão dupla (RV64IMFD)

- Adição: fadd.s
  Ex.: `fadd.s f0,f1,f2` `# f0=f1+f2`
- Subtração: fsub.s
  Ex.: `fsub.s f0,f1,f2` `# f0=f1-f2`
- Multiplicação: fmul.s
  Ex.: `fmul.s f0,f1,f2` `# f0=f1*f2`
- Divisão: fdiv.s
  Ex.: `fdiv.s f0,f1,f2` `# f0=f1/f2`
- Raiz Quadrada: fsqrt.s
  Ex.: `fsqrt.s f0,f1` `# f0=`$\sqrt{f1}$

Assembly RV32IM**F**

- Instruções de Comparação:

```
feq.s t0,f0,f1      # Se f0=f1 então t0=1 senão t0=0
flt.s t0,f0,f1      # Se f0<f1 então t0=1 senão t0=0
fle.s t0,f0,f1      # Se f0<=f1 então t0=1 senão t0=0
```

Logo o resultado da comparação é colocada em um registrador do Banco de Registradores principal e pode ser testado com `beq, bne, bge, btl`.

Assembly RV32IM**F**

- Instruções de Transferências de Dados

- Para/Da memória:

```
flw f0,Imm(t0)        # f0=Memoria[t0+Imm]
fsw f0,Imm(t0)        # Memoria[t0+Imm]=f0
```

- Para/Do Banco de Registradores principal

```
fmv.s.x f0,t0         # f0=t0    cópia dos bits
fmv.x.s t0,f0         # t0=f0    cópia dos bits
fcvt.s.w f0,t0        # f0=(float)t0  converte de inteiro para float
fcvt.w.s t0,f0        # t0=(int)f0    converte de float para inteiro
```

Exemplo:

```
float exemplo (float a, float b)
{
   if (a>b)
        return (a-b/a);
   else
        return (b+a*b);
}
```

Argumentos de entrada em fa0=a e fa1=b e saída em fa0.

```
exemplo: fle.s t0,fa0,fa1      # t0 = (a<=b)?1:0
         beq t0,zero,PULA      # t0==0? Logo (a>b)
         fmul.s fa0,fa0,fa1    # a=a*b
         fadd.s fa0,fa1,fa0    # a=b+a*b
         jalr x0,ra,0
PULA:    fdiv.s fa1,fa1,fa0    # b=b/a
         fsub.s fa0,fa0,fa1    # a=a-b/a
         jalr x0,ra,0
```