22.15 Electrónica V
Trabajo práctico 2: Procesador EV20

Chiocci, Ramiro Goytía, Agustín Kipen, Javier
Parra, Rocío Reina Kiperman, Gonzalo

Instituto Tecnológico de Buenos Aires

Martes 30 de junio de 2020

Contenido

Contenido

El procesador EV20

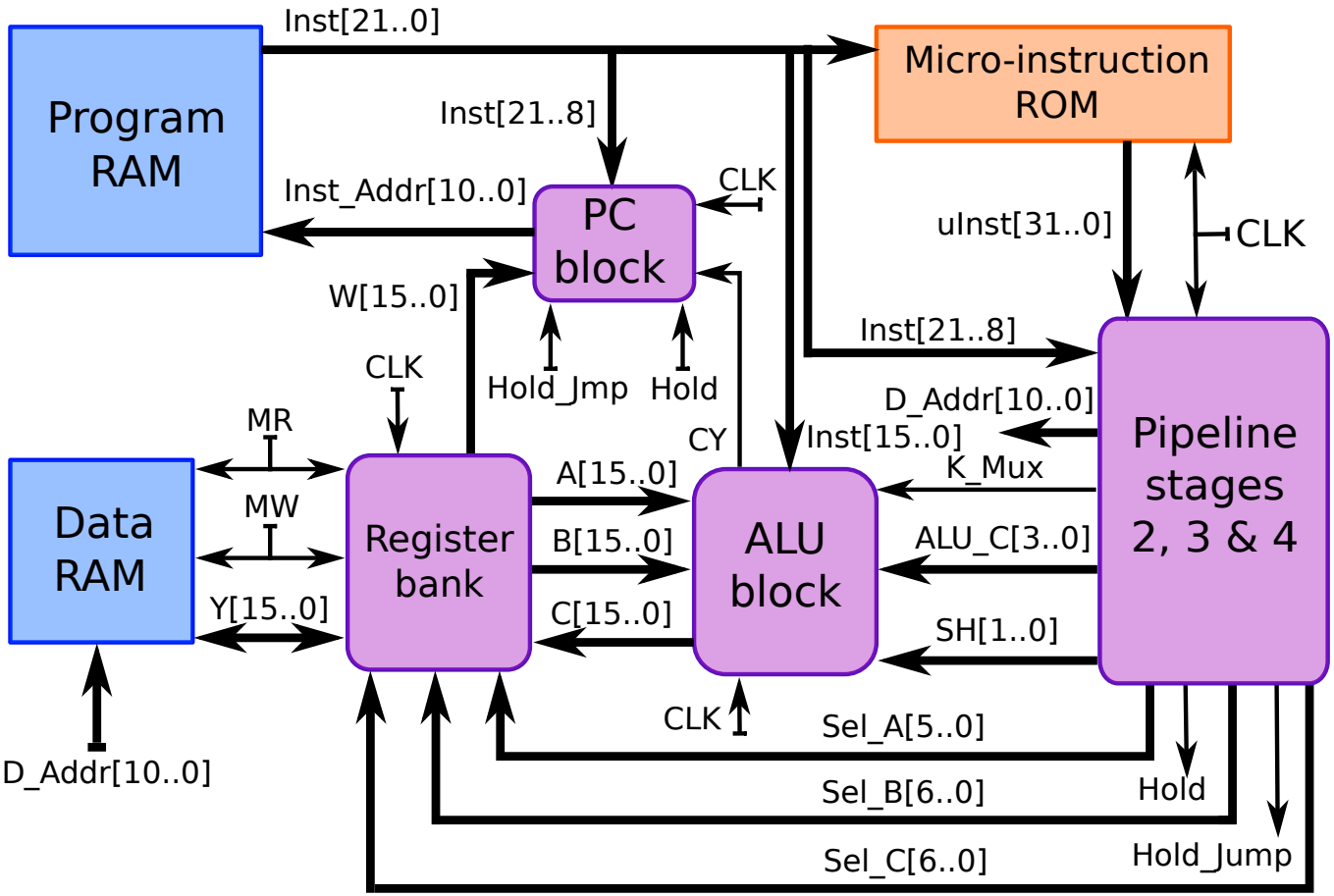

Figura: Diagrama de bloques del EV20

El procesador EV20

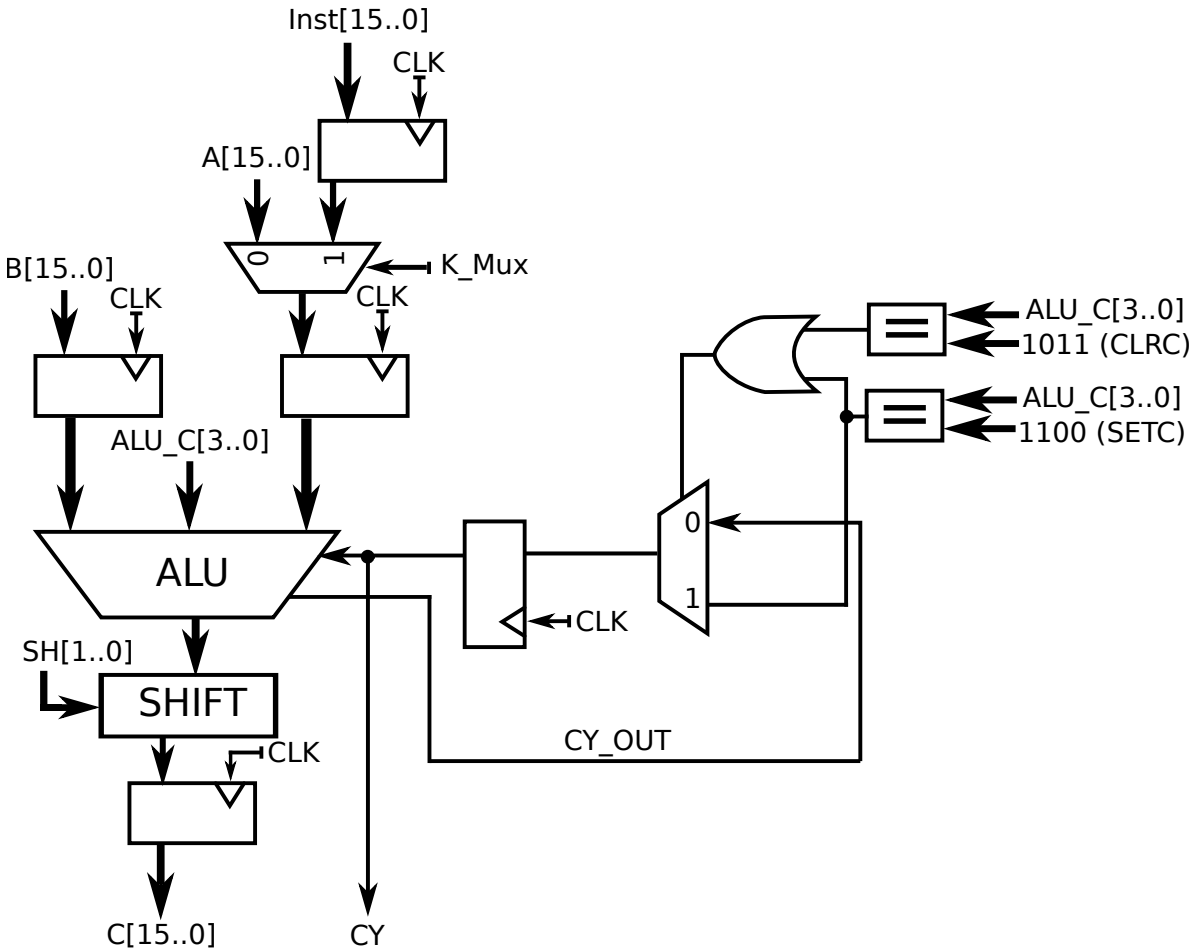

Figura: Bloque de control de la ALU

Contenido

Pipeline

- Etapa 1: fetch
  - instrucción
- Etapa 2: decode
  - selección de A y B
  - KMux
  - dirección de memoria
- Etapa 3: operand
  - ALU y shift control
  - control de memoria (MR, MW, I, i)
- Etapa 4: execute
  - selección de C
- Etapa 5: retire

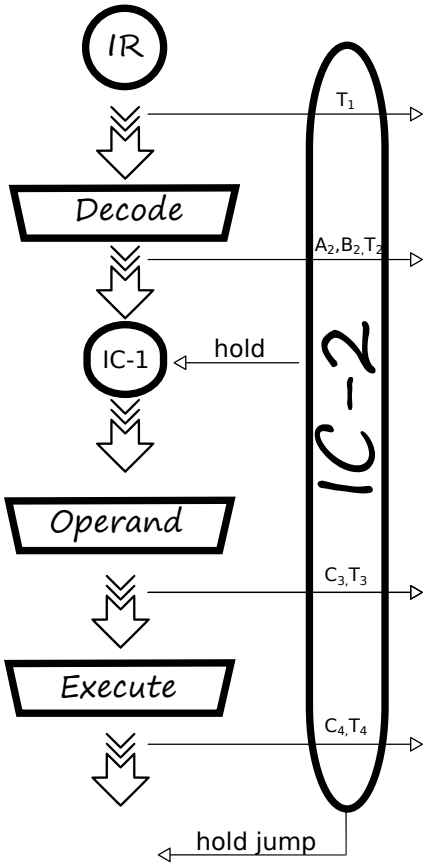

Figura: Control de dependencias

Pipeline

Notación del registro type:

$$T_i = \left[b^i_J, b^i_{CW}, b^i_{CR}, b^i_{RW}, b^i_{RR}, b^i_{WW}, b^i_{WR}\right]$$

Conflictos de dependencias:

- Dependencias RAW en el registro W y en el banco de registros
- Caso WAW con lectura de memoria
- Dependencias en los saltos

Contenido

Manejo de saltos

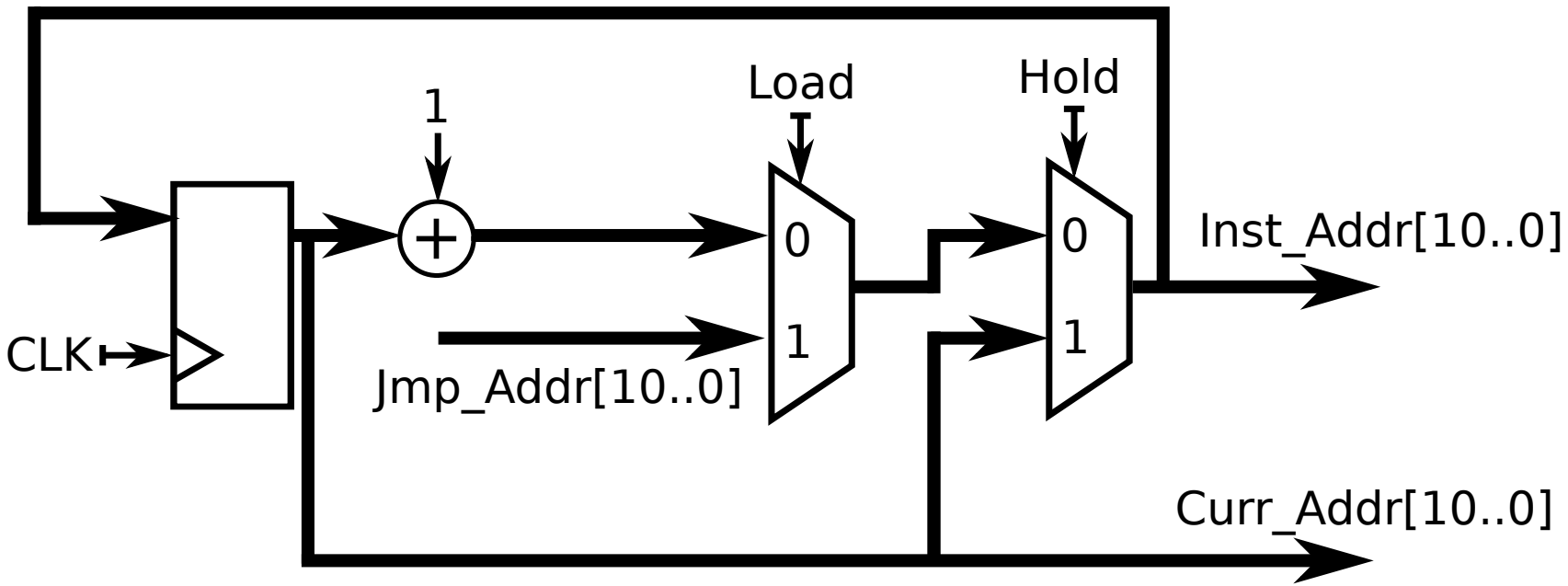

Figura: Bloque PC principal

Manejo de saltos

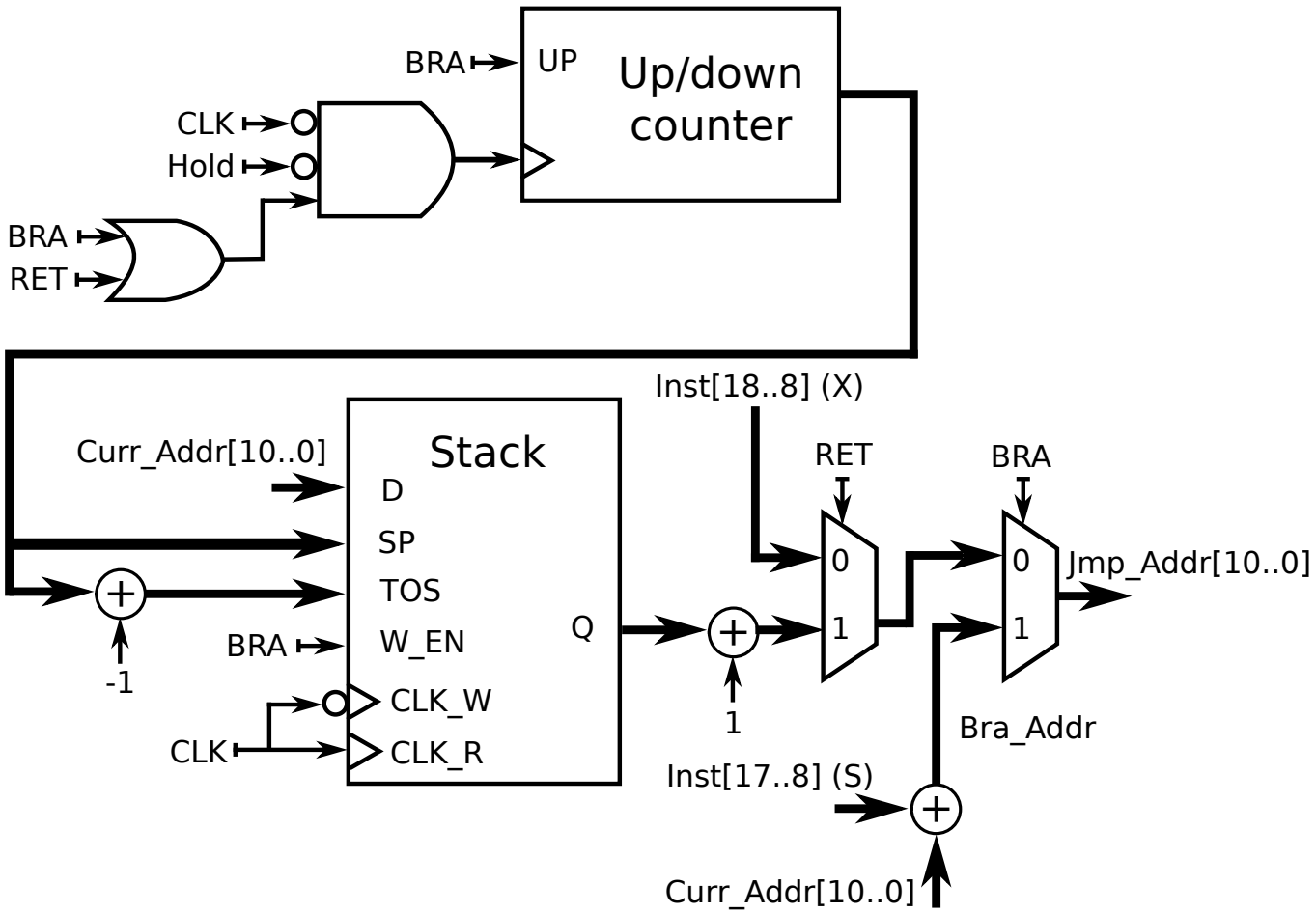

Figura: Bloque para control de saltos a subrutinas

Contenido

Modo de direccionamiento indexado

Motivación

- Acceso directo a memoria: cada dirección a la que se quiere acceder debe estar explícita en el código
- Engorroso para strings, arreglos
- Indexado: buena relación dificultad de implementación vs mejora obtenida

Implementación

- Dos registros base: R_0 y R_1
- Se ignoran los 4 bits más significativos para obtener direcciones de 11 bits

Modo de direccionamiento indexado: ISA

Cambios al ISA

- Sólo se modifican las instrucciones de acceso a memoria
- Se mantiene que mirando los primeros 7 bits de la instrucción se puede determinar la microinstrucción
- Se aprovecha que quedan 8 bits de don't care por utilizar operandos de 16 bits

Instrucción	ID	Dirección	Don't care
MOM W,Y	0101	yyy yyyy yyyy	dddd dddd
MOM Y,W	0100	yyy yyyy yyyy	dddd dddd

Tabla: Instrucciones de lectura y escritura del EV20 original

Modo de direccionamiento indexado: ISA

Cambios al ISA

- Bit para determinar modo de direccionamiento
- Bit para determinar registro base
- 6 bits de don't care en lugar de 8

Instrucción	ID	i	Dirección	Don't care
$MOM\ W, Y$	01010	d	yyy yyyy yyyy	dd dddd
$MOM\ Y, W$	01000	d	yyy yyyy yyyy	dd dddd
$MOM\ W, Y + R_i$	01011	i	yyy yyyy yyyy	dd dddd
$MOM\ Y + R_i, W$	01001	i	yyy yyyy yyyy	dd dddd

Tabla: Instrucciones de lectura y escritura del EV20 con modo de direccionamiento indexado

Modo de direccionamiento indexado: hardware

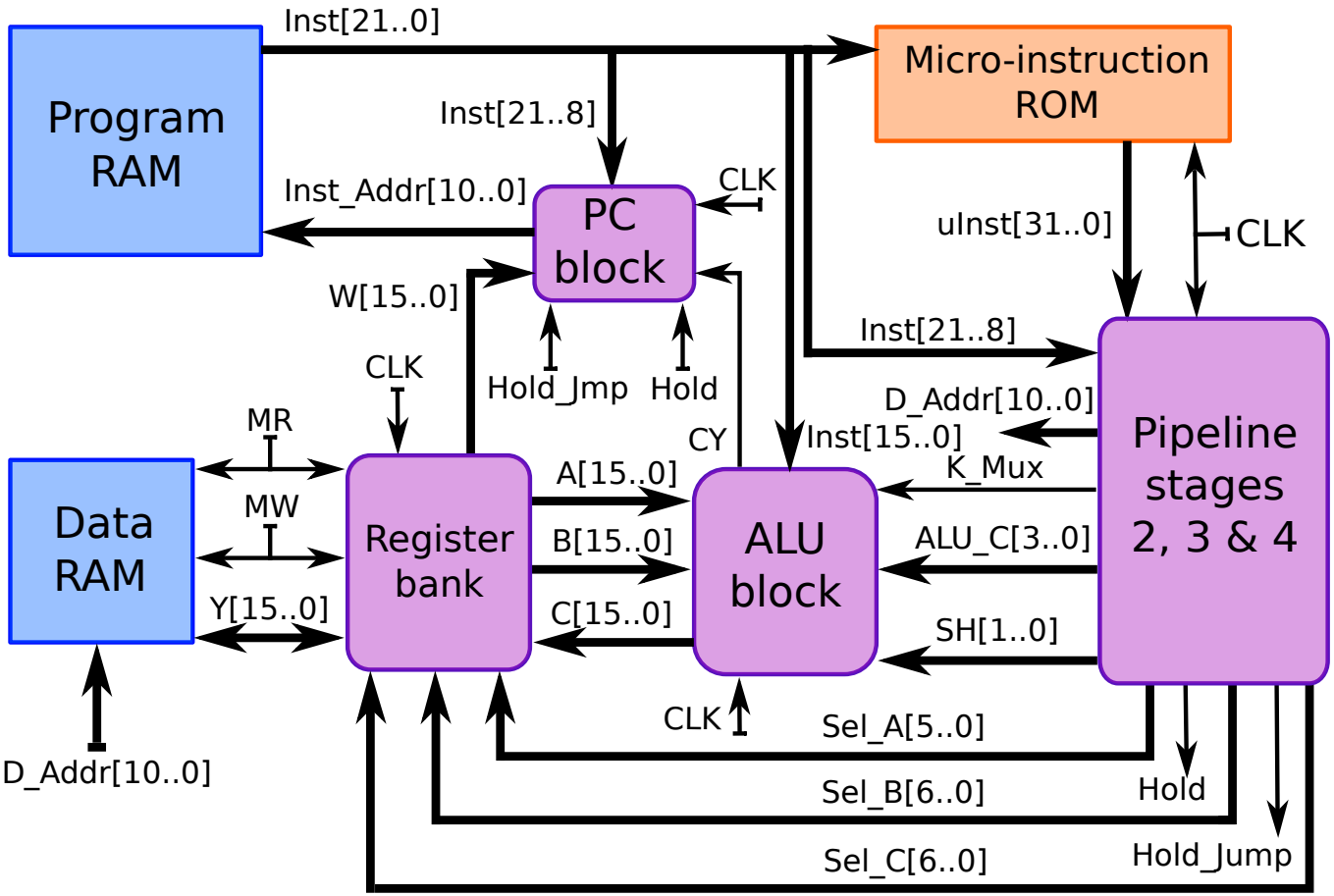

Figura: Diagrama de bloques del EV20 original

Modo de direccionamiento indexado: hardware

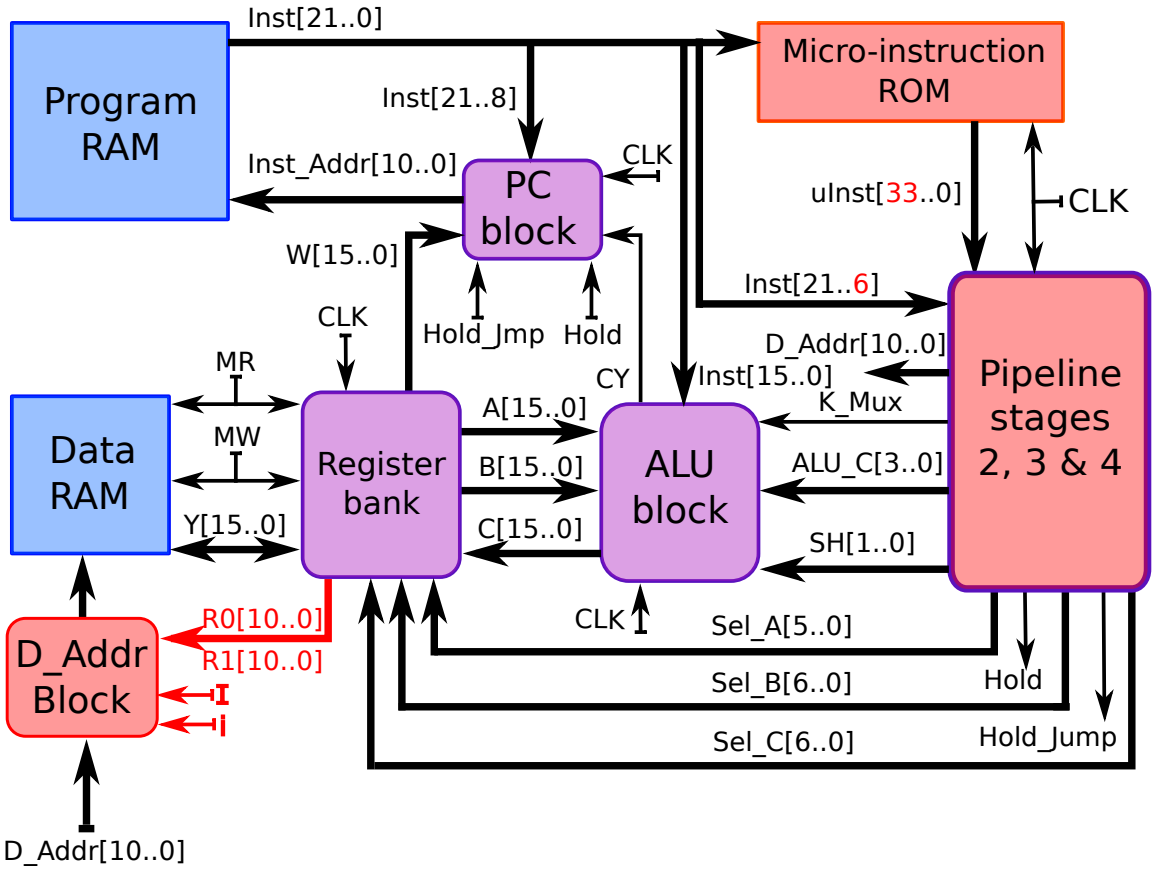

Figura: Diagrama de bloques del EV20 con modo de direccionamiento indexado

Modo de direccionamiento indexado: hardware

Modificaciones del hardware

- Micro-instruction ROM:
  - dos bits más (i e I) por microinstrucción, que se agregan al campo M
- Bloque D_Addr:
  - dos sumadores: $D\_Addr + R_0$ y $D\_Addr + R_1$
  - mux para elegir entre los dos sumadores (selector i)
  - mux para elegir entre D_Addr y $D\_Addr + R_i$ (selector I)
- Pipeline:
  - D_Addr se obtiene de *Inst[15..6]* en lugar de *Inst[17..8]*
  - UC1: donde recibía dos bits de M, ahora recibe 4
  - UC2: donde recibía dos bits de M, ahora recibe 4
    - control de dependencias RAW de R_0 y R_1, caso particular modo direccionamiento indexado

Contenido

Validación

Pruebas realizadas

- Pruebas modulares:
  - bloque ALU
  - pipeline
- Verificación de funcionamiento:
  - dependencias RAW en registro W
  - escritura y lectura a memoria, dependencia WAW en memoria
  - todas las instrucciones
  - saltos y subrutinas
  - modo de direccionamiento indexado

Muestra de funcionamiento: Fibonacci

Muchas gracias