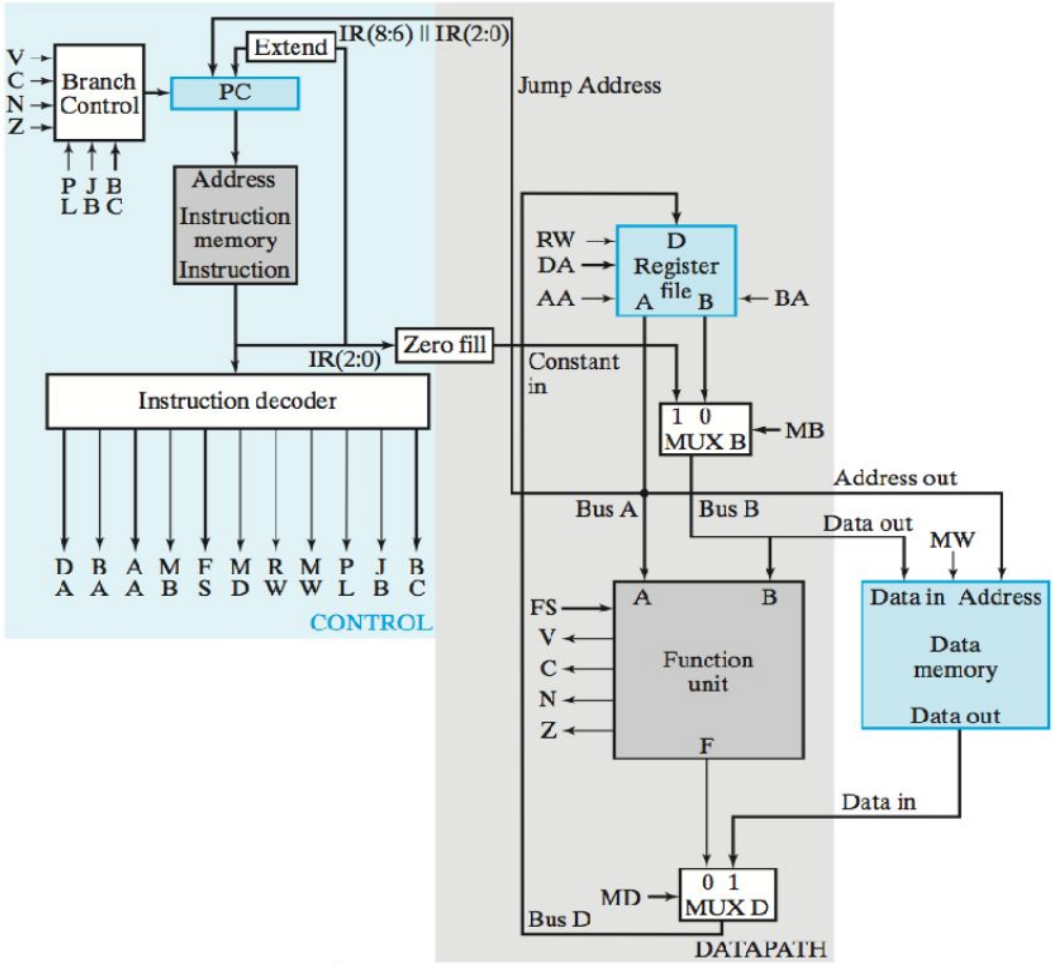
Ayudantia 10

IEE2713 - Sistemas Digitales

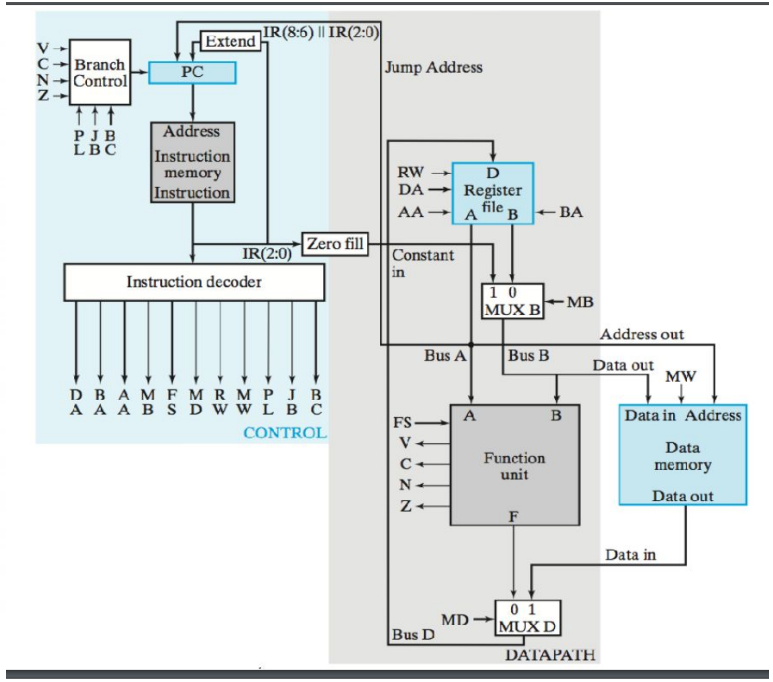
Diseñe un bloque secuencial que contenga el branch control y el program counter (PC) del computador sencillo visto en clase (y que tiene disponible en el esquema del material suplementario al enunciado). Considere además lo siguiente:

1. Puede utilizar todos los circuitos vistos en clase como cajas negras, por ejemplo: contadores, sumadores, multiplexores, decodificadores, etc.
2. Las entradas de su bloque son:
    - N: señal que indica que un número es igual a cero
    - Z: señal que indica que un número es negativo
    - AD: número binario que se suma al PC en caso de un BRZ o BRN
    - RA: número binario con el que se carga el PC en caso de un JMP
    - PL, JB y BC: los identificadores de tipo de operación para efectos del PC
    - **No considere en su diseño** las señales de entrada V ni C que aparecen en el esquema
3. La salida de su bloque es el número que indica el PC en cada flanco de clock


V
C
N
Z
Branch Control
PC
Extend
IR(8:6) || IR(2:0)
Jump Address
P L
J B
B C
Address
Instruction memory
Instruction
RW
DA
AA
D
Register file
A
B
BA
IR(2:0)
Zero fill
Constant in
Instruction decoder
1 0
MUX B
MB
Address out
Bus A
Bus B
Data out
MW
D A
B A
A A
M B
F S
M D
R W
M W
P L
J B
B C
CONTROL
FS
V
C
N
Z
A
B
Function unit
F
Data in Address
Data memory
Data out
Data in
MD
0 1
MUX D
Bus D
DATAPATH

Diseñe un bloque secuencial que contenga el branch control y el program counter (PC) del computador sencillo visto en clase (y que tiene disponible en el esquema del material suplementario al enunciado). Considere además lo siguiente:

1. Puede utilizar todos los circuitos vistos en clase como cajas negras, por ejemplo: contadores, sumadores, multiplexores, decodificadores, etc.
2. Las entradas de su bloque son:
   - N: señal que indica que un número es igual a cero
   - Z: señal que indica que un número es negativo
   - AD: número binario que se suma al PC en caso de un BRZ o BRN
   - RA: número binario con el que se carga el PC en caso de un JMP
   - PL, JB y BC: los identificadores de tipo de operación para efectos del PC
   - **No considere en su diseño** las señales de entrada V ni C que aparecen en el esquema
3. La salida de su bloque es el número que indica el PC en cada flanco de clock



Escriba en assembler un código que entregue los "N" primeros números de una progresión aritmética, partiendo desde el número "P", y luego guarde dichos números de la progresión aritmética en la memoria RAM del computador sencillo visto en clase.

Para esto considere lo siguiente:

1. Una progresión aritmética es una secuencia de números cuya diferencia "D" entre dos números consecutivos es constante.
2. Los parámetros "N", "D" y "P" son pequeños, de tal forma que todos los números de la secuencia solicitada caben en la memoria RAM de su computador sencillo.
3. "N" es un entero positivo que está guardado en la dirección 18 de la memoria RAM.
4. "P" es un entero positivo que está guardado en la dirección 23 de la memoria RAM.
5. "D" es un entero positivo que está guardado en la dirección 31 de la memoria RAM.
6. Los números de la progresión aritmética deben ser guardados en la memoria RAM en forma consecutiva a partir de la dirección 33.
7. Programe su código assembler considerando que pueda operar sin cambio alguno para distintos valores de "N", "D" y "P".
8. Considere que la unidad de registros se inicializará con **un reset en todos los registros**.
9. Considere todas las restricciones del computador sencillo vistas en clase (número de registros, número de bits disponibles, etc.)

Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move A	0000000	MOVA	RD, RA	R[DR] ← R[SA]*	N, Z
Increment	0000001	INC	RD, RA	R[DR] ← R[SA] + 1*	N, Z
Add	0000010	ADD	RD, RA, RB	R[DR] ← R[SA] + R[SB]*	N, Z
Subtract	0000101	SUB	RD, RA, RB	R[DR] ← R[SA] − R[SB]*	N, Z
Decrement	0000110	DEC	RD, RA	R[DR] ← R[SA] − 1*	N, Z
AND	0001000	AND	RD, RA, RB	R[DR] ← R[SA] ∧ R[SB]*	N, Z
OR	0001001	OR	RD, RA, RB	R[DR] ← R[SA] ∨ R[SB]*	N, Z
Exclusive OR	0001010	XOR	RD, RA, RB	R[DR] ← R[SA] ⊕ R[SB]*	N, Z
NOT	0001011	NOT	RD, RA	R[DR] ← $\overline{\text{R[SA]}}$ *	N, Z
Move B	0001100	MOVB	RD, RB	R[DR] ← R[SB]*	
Shift Right	0001101	SHR	RD, RB	R[DR] ← sr R[SB]*	
Shift Left	0001110	SHL	RD, RB	R[DR] ← sl R[SB]*	
Load Immediate	1001100	LDI	RD, OP	R[DR] ← zf OP*	
Add Immediate	1000010	ADI	RD, RA, OP	R[DR] ← R[SA] + zf OP*	N, Z
Load	0010000	LD	RD, RA	R[DR] ← M[SA]*	
Store	0100000	ST	RA, RB	M[SA] ← R[SB]*	
Branch on Zero	1100000	BRZ	RA, AD	if (R[SA] = 0) PC ← PC + se AD, if (R[SA] ≠ 0) PC ← PC + 1	N, Z
Branch on Negative	1100001	BRN	RA, AD	if (R[SA] < 0) PC ← PC + se AD, if (R[SA] ≥ 0) PC ← PC + 1	N, Z
Jump	1110000	JMP	RA	PC ← R[SA]	

* For all of these instructions, PC ← PC − 1 is also executed to prepare for the next cycle.

Escriba en assembler un código que entregue los "N" primeros números de una progresión aritmética, partiendo desde el número "P", y luego guarde dichos números de la progresión aritmética en la memoria RAM del computador sencillo visto en clase.

Para esto considere lo siguiente:

1. Una progresión aritmética es una secuencia de números cuya diferencia "D" entre dos números consecutivos es constante.
2. Los parámetros "N", "D" y "P" son pequeños, de tal forma que todos los números de la secuencia solicitada caben en la memoria RAM de su computador sencillo.
3. "N" es un entero positivo que está guardado en la dirección 18 de la memoria RAM.
4. "P" es un entero positivo que está guardado en la dirección 23 de la memoria RAM.
5. "D" es un entero positivo que está guardado en la dirección 31 de la memoria RAM.
6. Los números de la progresión aritmética deben ser guardados en la memoria RAM en forma consecutiva a partir de la dirección 33.
7. Programe su código assembler considerando que pueda operar sin cambio alguno para distintos valores de "N", "D" y "P".
8. Considere que la unidad de registros se inicializará con **un reset en todos los registros**.
9. Considere todas las restricciones del computador sencillo vistas en clase (número de registros, número de bits disponibles, etc.)

Instruction	Opcode	Mnemonic	Format	Description	Status Bits
Move A	0000000	MOVA	RD, RA	R[DR] ← R[SA]*	N, Z
Increment	0000001	INC	RD, RA	R[DR] ← R[SA] + 1*	N, Z
Add	0000010	ADD	RD, RA, RB	R[DR] ← R[SA] + R[SB]*	N, Z
Subtract	0000101	SUB	RD, RA, RB	R[DR] ← R[SA] − R[SB]*	N, Z
Decrement	0000110	DEC	RD, RA	R[DR] ← R[SA] − 1*	N, Z
AND	0001000	AND	RD, RA, RB	R[DR] ← R[SA] ∧ R[SB]*	N, Z
OR	0001001	OR	RD, RA, RB	R[DR] ← R[SA] ∨ R[SB]*	N, Z
Exclusive OR	0001010	XOR	RD, RA, RB	R[DR] ← R[SA] ⊕ R[SB]*	N, Z
NOT	0001011	NOT	RD, RA	R[DR] ← $\overline{R[SA]}$ *	N, Z
Move B	0001100	MOVB	RD, RB	R[DR] ← R[SB]*	
Shift Right	0001101	SHR	RD, RB	R[DR] ← sr R[SB]*	
Shift Left	0001110	SHL	RD, RB	R[DR] ← sl R[SB]*	
Load Immediate	1001100	LDI	RD, OP	R[DR] ← zf OP*	
Add Immediate	1000010	ADI	RD, RA, OP	R[DR] ← R[SA] + zf OP*	N, Z
Load	0010000	LD	RD, RA	R[DR] ← M[SA]*	
Store	0100000	ST	RA, RB	M[SA] ← R[SB]*	
Branch on Zero	1100000	BRZ	RA, AD	if (R[SA] = 0) PC ← PC + se AD, if (R[SA] ≠ 0) PC ← PC + 1	N, Z
Branch on Negative	1100001	BRN	RA, AD	if (R[SA] < 0) PC ← PC + se AD, if (R[SA] ≥ 0) PC ← PC + 1	N, Z
Jump	1110000	JMP	RA	PC ← R[SA]	

* For all of these instructions, PC ← PC − 1 is also executed to prepare for the next cycle.